НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Автоматики та управління в технічних системах

«До захисту допущено»
Завідувач кафедри
_____ О.І. Ролік
«___»_____20__р.

Дипломний проєкт

на здобуття ступеня бакалавра
з напряму підготовки 151 «Автоматизація та
комп'ютерно-інтегровані технології»
на тему «Система дистанційної інсталяції програмного забезпечення»

Виконав:
Студент IV курсу, групи ІА-62
Теленик Андрій Миколайович _____

Керівник:
асистент
Вовк Євгеній Володимирович _____

Рецензент
Доцент кафедри АУТС, к.т.н., доцент
Жданова Олена Григорівна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»
Освітньо-професійна програма «Комп'ютеризовані системи управління»

«Затверджую»
Завідувач кафедри
_____ О.І. Ролік
«___»_____20__р.

ЗАВДАННЯ
на дипломний проєкт студенту
Теленику Андрію Миколайовичу

1. Тема проєкту «Система дистанційної інсталяції програмного забезпечення», керівник проєкту Вовк Євгеній Андрійович, асистент, затверджені наказом по університету від «___»_____20__р. №_____

2. Термін подання студентом проєкту 09.06.2020

3. Вихідні дані до проєкту: Система дистанційної інсталяції програмного забезпечення

4. Зміст пояснювальної записки: огляд предметної області та існуючих рішень. Аналіз та вибір архітектури додатку. Аналіз механізмів рішення питання побудови подібних систем. Аналіз та вибір технології реалізації. Детальний опис архітектури. Опис програмного рішення .

5. Перелік графічного матеріалу: графічне порівняння характеристик різних технологій. Графік порівняння залежності швидкодії існуючих систем від кількості абонентів мережі. Графік зростання вартості системи в залежності від розміру підприємства. .

6. Консультанти розділів проєкту

Розділ	Прізвище, ім’я, по батькові консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання_____

Календарний план

№	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка

Студент А.М. Теленик

Керівник Є.А. Вовк

АНОТАЦІЯ

Теленик А.М. Система дистанційної інсталяції програмного забезпечення. КПІ ім. Ігоря Сікорського, Київ, 2020.

Проєкт містить 64 с. тексту, 16 рисунків, посилання на 10 літературних джерел та 6 онлайн-ресурсів.

Ключові слова: розгортання програмного забезпечення; IT-інфраструктура; локальна мережа; апгрейд програм; інтерфейс; тиха інсталяція.

Предметом дослідження є система дистанційної інсталяції програмного забезпечення.

Метою дипломного проєкту є підвищення ефективності та зниження вартості роботи системи.

У дипломному проєкті розроблено систему дистанційної інсталяції програмного забезпечення, що забезпечує розгортання програмного забезпечення на комп’ютери у межах корпоративної локальної мережі. Проведено аналіз предметної області та існуючих рішень і технологій, внаслідок чого зроблено висновок про необхідність обрання клієнт-серверної архітектури. Застосунок обраної архітектури реалізовано засобами .NET, мови програмування C#.

Отримані результати можуть бути корисними при автоматизації аналогічних чи подібних об’єктів.

ANNOTATION

Telenyk A.M. Remote software installation system. Igor Sikorsky Kyiv Polytechnic institute, Kyiv, 2020.

The project contains 64 pages of text, 16 figures, links to 10 literary sources and 6 online resources.

Keywords: software deployment; IT-infrastructure; local network; software upgrade; interface; quiet installation.

The subject of research is a system of remote software installation.

The purpose of the diploma project is to increase the efficiency and cost of the system.

The diploma project developed a system of remote software installation, which provides software deployment to computers within the corporate local network. The analysis of the subject area and existing solutions and technologies is carried out, as a result of which the conclusion about the necessity of choosing the client-server architecture is made. The application of the selected architecture is implemented by means of .NET, C # programming language.

The results obtained can be useful in automation of similar objects.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	**ІА62.250БАК.002 ПЗ**	Пояснювальна записка	99		
6						
7	А3	**ІА62.250БАК.003 ОВ**	Графік порівняння швидкодії систем	1		
8						
9	А3	**ІА62.250БАК.004 ОВ**	Графік ефективності тенологій	1		
10						
11	А3	**ІА62.250БАК.006 ОВ**	Схема Юзкейс	1		
12						
13	А3					
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	**ІА62.250БАК.001 ТП**		
Розроб.		Теленик			**Відомість технічного проекту**	Літ.	Аркуш	Аркушів
Перевір.		Вовк				Т	1	1
Реценз.					**Система дистанційної інсталяції програмного забезпечення**	НТУУ «КПІ ім. І. Сікорського» ФІОТ група ІА-62		
Н. Контр.								
Затв.								

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

НТР – науково-технічна революція.

SOA – сервер-орієнтована архітектура.

WOA – веб-орієнтована архітектура.

CLR – мовно-командне середовище.

UAC – контроль доступу користувача.

ОС – Операційна Система

ВСТУП

З розгортанням нового етапу Науково-Технічної Революції (НТР) відбувається постійне зростання кількості необхідних у науковій, виробничій, навчальній та інших сферах програмних застосунків (продуктів) та різноманіття їх форм. Від файлових менеджерів до графічних редакторів, усе більше і більше програмних застосунків допомогають людям практично усіх професій, у діяльності яких використовується цифрова техніка. Особливо це різноманіття програм стає у пригоді студентам та працівникам ІТ-сфери, значно полегшуючи процес навчання або роботи та прискорюючи їх.

Одночасно зі збільшенням різноманіття доступних програмних застосунків відбувається усе більш інтенсивний процес консолідації (кластеризації) комп’ютерів. Цей процес полягає у поступовій інтеграції комп’ютерів (та інших комп’ютерних пристроїв – серверів, маршрутизаторів, принтерів та ін.) у склад локальних мереж, які, в свою чергу, об’єднуються в одну пан-світову комп’ютерну мережу (Інтернет). Загальносвітова тенденція до консолідації електронних пристроїв проявляється у впроваджені локальних мереж фірм, університетських кафедр, виробничих підприємств, військових частин. Навіть в побуті збільшується використання комп’ютерних мереж для відпочинку або спрощення доступу до електронних благ цивілізації.

Коли комп'ютери були надзвичайно великими, дорогими та об'ємними (мейнфрейми та мінікомп'ютери), програмне забезпечення часто постачалося разом із обладнанням виробниками. Якщо бізнес-програмне забезпечення потрібно встановити на існуючий комп'ютер, це може зажадати дорогого, трудомісткого відвідування системного архітектора чи консультанта. Для складної локальної інсталяції корпоративного програмного забезпечення сьогодні це може бути значно скорочено.

Поширення Інтернету зробило можливим комплексне розроблення програмного забезпечення.

Це і зумовило необхідність виникнення засобу, який б уможливив швидке забезпечення усіх абонентів комп’ютерних мереж різноманітними програмними додатками – системи дистанційної інсталяції програмного забезпечення, яка могла б забезпечувати проведення стосовно програмного забезпечення, що розгортається у рамках локальної мережі, наступних процедур:

– реліз - реліз випливає із завершеного процесу розробки, а іноді класифікується як частина процесу розробки, а не процес розгортання. Він включає всі операції з підготовки системи до складання та передачі в комп'ютерну систему (и), на якій вона буде запускатися у виробництві. Отже, іноді це передбачає визначення ресурсів, необхідних для роботи системи з дозволеною продуктивністю та плануванням та / або документування подальших дій процесу розгортання;

– інсталяція та активація програми - для простих систем інсталяція передбачає встановлення певної форми команд, ярликів, сценарію або послуги для виконання програмного забезпечення (вручну або автоматично). Для складних систем це може включати конфігурацію системи - можливо, задаючи питанням кінцевого користувача про її передбачуване використання або безпосередньо запитуючи їх, як вони хотіли б, щоб вона була налаштована - та / або зробила всі необхідні підсистеми готовими до використання. Активація - це діяльність з першого запуску виконуваного компонента програмного забезпечення (не плутати з загальним використанням терміну активація щодо ліцензії на програмне забезпечення, що є функцією систем управління цифровими правами.) У більш великих розгортаннях програмного забезпечення на серверах основна копія програмного забезпечення, яке використовується користувачами, -

"виробнича" - може бути встановлена на виробничому сервері у виробничому середовищі. Інші версії розгорнутого програмного забезпечення можуть бути встановлені в тестовому середовищі, середовищі розробки та середовищі відновлення після аварій. У складних середовищах безперервної доставки та / або програмному забезпеченні як сервісних систем, різні конфігуровані версії системи можуть навіть існувати одночасно у виробничому середовищі для різних внутрішніх або зовнішніх клієнтів (це відома як архітектура для багатьох орендарів) або навіть бути поступово розгортаються паралельно для різних груп клієнтів, з можливістю скасування одного або декількох паралельних розгортань. Наприклад, Twitter, як відомо, використовує останній підхід для тестування нових функцій та змін інтерфейсу користувача. Групу "прихованого живого" також можна створити у виробничому середовищі, що складається з серверів, які ще не підключені до балансування виробничого навантаження для цілей розгортання;

– деактивація - це зворотна активація, і стосується вимкнення будь-яких вже виконуваних компонентів системи. Деактивація часто потрібна для виконання інших заходів з розгортання, наприклад, програмне забезпечення може знадобитися деактивувати до того, як можна буде виконати оновлення. Практику видалення з експлуатації нечасто використовуваних або застарілих систем часто називають відстороненням програми або виведенням з експлуатації додатків;

– видалення - зворотна установка. Саме видалення системи вже не потрібно. Це може також включати деяку переконфігурацію інших програмних систем для усунення залежностей від видаленої системи;

– оновлення - процес оновлення замінює попередню версію всієї або частини програмної системи на новіший випуск. Зазвичай він складається з дезактивації з подальшим встановленням. У деяких системах,

таких як Linux, коли використовується менеджер пакунків системи, стара версія програмного додатку зазвичай також видаляється як автоматична частина процесу. (Це пов’язано з тим, що менеджери пакетів Linux, як правило, не підтримують установку декількох версій програмного додатку одночасно, якщо тільки програмний пакет не був розроблений спеціально для усунення цього обмеження);

– вбудоване оновлення - механізми встановлення оновлень вбудовані в деякі програмні системи (або, у випадку деяких операційних систем, таких як Linux, Android та iOS, в саму операційну систему). Автоматизація цих процесів оновлення коливається від повністю автоматичного до ініційованого та керованого користувачем. Norton Internet Security - це приклад системи з напівавтоматичним методом отримання та встановлення оновлень як антивірусних визначень, так і інших компонентів системи. Інші програмні продукти надають механізми запитів для визначення наявності оновлень;

– відстеження версій - системи відстеження версій допомагають користувачеві знаходити та встановлювати оновлення програмних систем. Наприклад: Каталог програмного забезпечення зберігає версію та іншу інформацію для кожного програмного пакету, встановленого в локальній системі. Одне клацання кнопки запускає вікно браузера на веб-сторінку оновлення програми, включаючи автоматичне заповнення імені користувача та пароля для сайтів, які потребують входу. У Linux, Android та iOS цей процес ще простіший, оскільки в операційну систему вбудований стандартизований процес відстеження версій (для програмних пакетів, встановлених офіційно підтримуваним способом), тому не потрібно проводити окремі кроки входу, завантаження та виконання - тому процес може бути налаштований повністю автоматизовано. Деякі сторонні

програми також підтримують автоматичне відстеження версій та оновлення певних програмних пакетів Windows.

Отже, виникає потреба у розробка системи дистанційної інсталяції програмного забезпечення. Основними задачами, які призначена вирішувати система, є виконання необхідних операцій зі встановлення програмного забезпечення на комп’ютери, об’єднані в мережу підприємства, а саме: встановлення та підтримання з’єднання між сервером та клієнтами, вибір та архівація необхідних для інсталяції програмних компонент, передача сформованого таким чином архіву по мережі, перевірка правильності його отримання та розпаковка.

Областю застосування даного продукту має бути підтримка роботи підприємств, у діяльності яких велику роль грає програмне забезпечення – офісів, ІТ-компаній, навчальних закладів, тощо. Система повинна відповідати наступним вимогам:

1) використання в локальних мережах;
2) мережа в рамках підприємства;
3) наявність одного серверу та багатьох підпорядкованих комп'ютерів;
4) зв'язок «один до багатьох»;
5) передача програмних компонентів у формі архіву;
6) можливість обирати файли різних форматів для додавання в архів;
7) верифікація правильності переданих даних;
8) незалежність системи від кількості абонентів локальної мережі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Загальні відомості

Вирішення питання централізованого оновлення програмного забезпечення на комп'ютерах підприємства можливе декількома шляхами:

Перший, найпростіший, шлях полягає у встановленні ПЗ на комп'ютери-абоненти вручну з фізичного носія – гнучкого диску, flash-накопичувача, ін. У такому випадку виключена будь-яка помилка мережі через фактичне незастосування мережевого підключення між комп'ютерами для процесу інсталяції. До того ж, адміністратор що виконує установку вручну може оперативно вирішити проблеми інсталяції по мірі їх виникнення. Разом з тим, переваги цього методу не можуть переважити численні недоліки – інсталяція при такому варіанті займає значний час, що збільшується пропорційно кількості комп'ютерів у мережі. Необхідність виконувати монотонну та відповідальну роботу з інсталяції накладає на адміністратора велике навантаження, що призводить до втоми та втрати продуктивності праці. До того ж, вірогідність людської помилки при такому методі доволі висока – адміністратор може забути виконати інсталяцію на один чи навіть декілька комп'ютерів, або навіть виконати її неправильно.

Другий варіант вирішення проблеми полягає у створенні хмарного сховища, у якому адміністратор розміщає стиснуті дані необхідних для інсталяції програм та їх компонентів. Після розміщення усіх матеріалів у «хмарі» користувачі-абоненти мережі завантажують інсталяційні файли та самостійно встановлюють програмне забезпечення на свої машини. Переваги даного методу прямо протилежні недолікам попереднього – від адміністратора вимагається лише створити повний пакет необхідних програм та завантажити його у сховище, подальша інсталяція проводиться

операторами комп’ютерів-абонентів. Але навіть у цьому випадку є «підводне каміння» - такий метод містить вразливості відразу на мережевому, прикладному та транспортному рівнях взаємодії членів мережі. Передача даних з хмарного сховища може постраждати через дію різноманітних завад, від білого шуму до електромагнітних хвиль, розриву з’єднання або помилки пакетування TCP\IP. Окрім того, віддача інсталяції на відкуп операторам клієнтів не гарантує правильного встановлення софту – рівень їх досвіду та компетенції може не корелювати з рівнем адміністратора.

Нарешті, третій шлях, наскільки це можливо, компенсує недоліки першого та другого методів. Він полягає у використанні автоматизованої системи мережевої інсталяції – спеціальної програми, яка у режимі прямого підключення до клієнтів по локальній мережі буде проводити встановлення необхідного програмного забезпечення незалежно від наявності чи відсутності контролю з боку адміністратора. Така методика поєднує переваги першого та другого методів – компетенція адміністратора може забезпечувати інсталяцію, але у той же час він економить робочий час системи як при методі №2.

Така система повинна забезпечувати стійку роботу з усіма абонентами локальної мережі, допускаючи можливість працювати як з окремими комп’ютерами так і з цілими виділеними робочими групами комп’ютерів. По-друге, гіпотетична система інсталяції повинна мати можливість доступу до встановлених на клієнтських комп’ютерах програмних додатків щоб уникнути повторної інсталяції та, як її наслідок, нераціонального використання пам’яті комп’ютерів-абонентів, у випадку якщо необхідний софт все встановлений. Також очевидно, що необхідність швидкої встановки та оновлення програмного забезпечення для всіх абонентів мережі підприємства диктує потребу у можливості інсталяції з

готових пакетів софту, які будуть завантажені по мережі за допомогою протоколу TCP/IP у сумісному з операційною системою клієнтів вигляді. Усе це буде потребувати тісної роботи з мережевим, транспортним та презентаційним рівнями мережевої моделі. І, врешті-решт, не можна забувати про критерій простоти використання та обслуговування даної системи – робота з нею не повинна викликати проблем у непідготовленого користувача, а інтерфейс має бути простим та зрозумілим, реалізуючи принцип user-friendly.

1.2 Огляд існуючих рішень

Наразі існує дві програми з дистанційної інсталяції програмного забезпечення, що відповідають цим вимогам – *Total Software Deployment* та *Smart AutoInstall.*

1.2.1. програма Total Software Deployment (TSD).

Однією з найбільш досконалих з наявних систем автоматизованої дистанційної інсталяції є програма *Total Software Deployment (TSD)* - програма розгортання програмного забезпечення для ОС Windows, розроблена компанією Softinventive Lab. Total Software Deployment працює з двома незалежними базами даних: одна для програмного забезпечення та одна для комп'ютерів під керуванням Windows, наявних у мережі користувача. Програма може сканувати мережу для комп'ютерів Windows, щоб заповнити мережеву базу даних. Доступні три методи розгортання програмного забезпечення: Silent, Macro і SysShot. Розгортання здійснюється на комп'ютери, що зберігаються в мережевій базі даних за допомогою протоколу SMB.

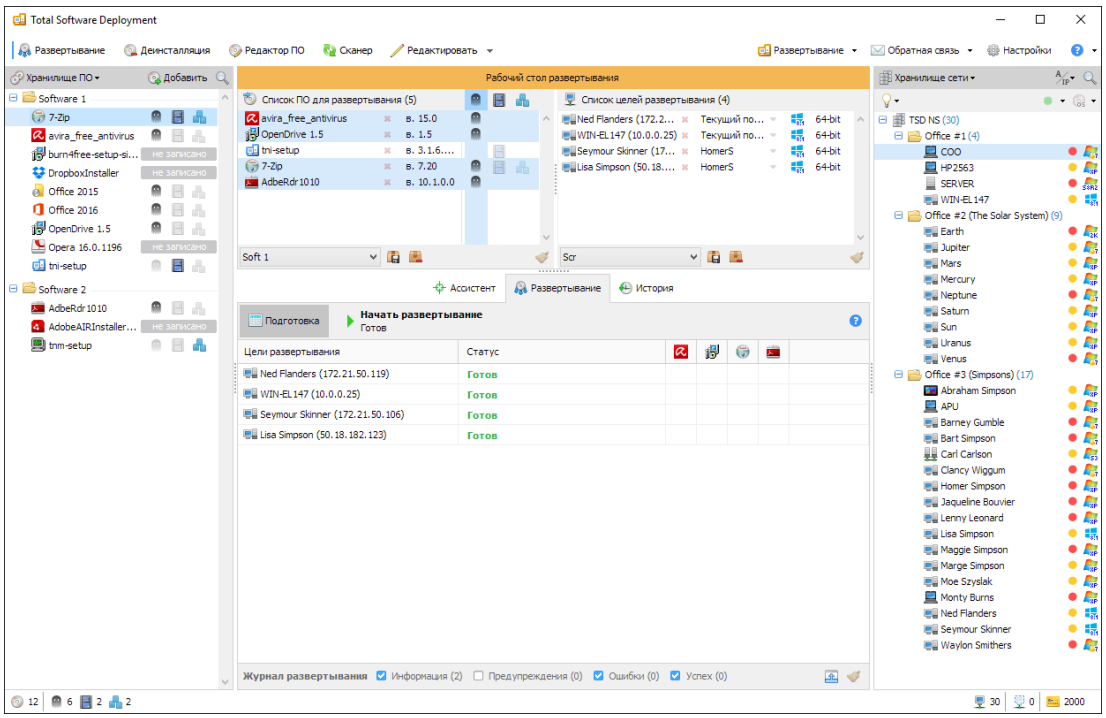

Рисунок 1.1. Інтерфейс програми Total Software Deployment

Ця програма призначена для управління та розгортання ПЗ в корпоративних мережах. Програмний додаток забезпечує виконання поставлених цілей шляхом сканування локальної мережі на предмет підключених до неї абонентів, перевірки наявності на них необхідного корпоративного програмного забезпечення, перевірки версії цього ПЗ та проведення апгрейду (або встановлення «з нуля») забезпечення згідно з результатом сканування. Потрібно відзначити наявність у *TSD* можливості проводити сканування не лише за конкретними IP-адресами окремих комп'ютерів, а й за робочими групами абонентів. Також програма надає можливість автоматично проводити контроль програмного забезпечення за заданими критеріями, або на россуд адміністратора працювати у ручному режимі, напрямую підключаючись до потрібного абонента та даючи команду на інсталяцію необхідного софту.

Автоматичний режим дозволяє визначати політики контролю софту для абонентів та особисто формувати пакети розгортування для тієї чи іншої

політики. У даному варіанті програма підготовлює інсталяційний пакет на основі доступу до операційної системи клієнта та реєстрів цієї ОС. Одночасно з швидкодією та високою точністю, даний метод має свої недоліки – так, доступ до реєстрів системи може призвести до небажаних наслідків для безпеки та конфіденційності. На додачу, такий спосіб можна використовувати лише для інсталяції відносно примітивного програмного забезпечення, яке не вимагає встановки драйверів, бібліотек та ін.

Total Software Deployment використовує технологію сканування з програмного продукту Total Network Inventory 3. Однак, на відміну від TNI 3, *TSD* надає інформацію тільки з категорії встановлених програм. Ця інформація може використовуватися для визначення необхідності розгортання програм на віддалений комп'ютер. Вся інша інформація виходить при скануванні для забезпечення кросспрограммного використання сховища мережі. Налаштувавши програми на одне і те ж сховище і зробивши сканування в одній з програм, вам не потрібно буде виконувати сканування в іншій програмі. Сховище буде актуальним на момент запуску будь-якої з програм.

Інструментарій управління Windows (Windows Management Instrumentation, WMI) - реалізація стандарту Web-Based Enterprise Management (WBEM) компанії Microsoft для операційних систем Microsoft Windows. WMI являє собою стандартний набір інтерфейсів доступу до пристроїв, додатків і параметрам операційної системи Windows. Використовуючи технологію WMI, *TSD* отримує дані про апаратне та програмне забезпечення, а також дані з реєстру комп'ютерів.

Програма *Total Software Deployment* дозволяє проводити інсталяцію в трьох режимах: «Macro», “SysShot” та “Silent”. Кожен з цих режимів має свої особливості.

Режим «Macro», тобто «Макро-інсталяція», усе програмне забезпечення, яке планується до дистанційної інсталяції, повинне бути встановлене на комп'ютері адміністратора мережі перед формуванням інсталяційного пакету. При цьому *TSD* «запам'ятовує» усі дії з інсталяції. Після цього відбувається формування інсталяційного пакету, у який разом з компонентами програми додається записана послідовність дій, та його відправка на комп'ютери – абоненти. По отриманню інсталяція відбувається автоматично за сформованою послідовністю дій. У даному способі мається на увазі застосування макросу. В цьому випадку виконується запис всіх потрібних маніпуляцій користувача з подальшим її відтворенням в процесі інсталяції. Даний підхід в більшості випадків застосовується для перекладу будь-яких дій, які регулярно виконуються, в автоматичний режим. В інтернеті можна знайти спеціальні додатки, які працюють за таким принципом.

Перевагою такого режиму є практична відсутність навантаження як на адміністратора, так і на користувача комп'ютера-клієнта, а також висока швидкість виконання інсталяції завдяки можливості її виконання паралельно на усіх комп'ютерах підприємства одночасно. Разом з тим, цей режим має і великі недоліки – він повністю не надає ніякої можливості коректувати інсталяцію, реагуючи на виникаючі помилки через особливості комп'ютера-клієнта. Також даний режим висуває великі вимоги до точності «первинної» інсталяції на комп'ютер адміністратора – будь-яка помилка під час цієї операції буде записана та відтворена у ході інсталяції на абоненти мережі.

Режим «SysShot», тобто «Системний знімок», також вимагає попередньої «контрольної» інсталяції на серверний комп'ютер адміністратора. Але, на відміну від «макро-інсталяції», система робить знімки (snapshot) ОС станом на безпосередньо перед початком інсталяції та

безпосередньо після неї. При формуванні інсталяційного пакету ці знімки порівнюються, різниця між ними архівується та використовується при інсталяції на комп’ютери – абоненти мережі. Особливістю даного режиму є велика залежність від конкретної операційної системи – тому він використовується переважно в системах з максимально близькими ОС на комп’ютерах, аж до рівня однакових збірок систем. Варто знати, що даний метод досить небезпечний і може завдати шкоди системі, тому використовувати його можуть тільки досвідчені користувачі. Крім цього, розглянутий спосіб варто використовувати тільки для примітивного софту, при інсталяції якого не коригуються драйвера, системні бібліотеки та інше.

Чи не найзручнішим аспектом *TSD* є використання так званої «тихої інсталяції» (“Silent”) - способу розгортання (deployment) програмних файлів, які потрібно інсталювати на комп’ютері не перетинаючись з поточною діяльністю оператора комп’ютера-абонента, тобто процес під’єднання та інсталяції йде, керований виключно з комп’ютера-сервера без виникнення зайвих вікон, що можуть відволікти клієнта.

Більшість сучасних інсталяційних пакетів підтримують режим тихої інсталяції. В цьому режимі програмне забезпечення встановлюється на комп'ютер без взаємодії з користувачем, виконуючи всі процеси автоматично.

Використання даного режиму передбачено (в більшості випадків) додаванням параметрів до командного рядка виконуваного файлу. Може знадобиться задати декілька параметрів для досягнення необхідного результату. Тиха інсталяція є найбільш рекомендованим методом для використання.

TSD дає можливість обирати режим «тихої інсталяції» через командну строку за допомогою набору потрібних параметрів, для чого не потрібно навіть запитувати дозволу від користувача клієнтського комп’ютера. Цей

режим має велику перевагу над двома попередніми у тому, що він дозволяє корегувати можливі помилки, що можуть виникнути у ході інсталяції, не відволікаючи оператора ПК від вирішення поточних задач.

Програма *TSD* має розвинений інтерфейс користувача, за допомогою якого адміністратор мережі має змогу обирати різноманітні функції програми. Інтерфейс показує наявні варіанти програмних додатків, які можна інсталювати та список абонентів мережі, до якої під'єднаний комп'ютер.

В якості додаткового функціоналу *TSD* можна відмітити можливість редагувати вже встановлене програмне забезпечення та проводити його деінсталяцію, що реалізується засобами, аналогічними віддаленій інсталяції.

Економічно, *TSD* можна віднести до сфери trialware – типу пропієтарного програмного забезпечення, використання якого є безкоштовним протягом 60 діб. Окрім цього, є додаткові обмеження випробувальної версії – максимальна кількість абонентів мережі, з якими можлива робота, становить 50, а одночасно можна розгортати лише 5 пакетів. Після закінчення даного випробувального терміну користувач повинен купляти ліцензійну програму. Вартість програми залежить від кількості абонентів у локальній мережі, яку буде обслуговувати система – ціна падає в залежності від неї, якщо версія для мережі з 25 абонентами коштує $90 (3 долари 60 центів за один комп’ютер-клієнт), то версія для мережі з 2000 абонентами коштує $1790 (90 центів за один комп’ютер-клієнт). Також покупець може обрати версію з безлімітною кількістю абонентів, вона коштує $2490. Оплата програми можлива через PayPal, Wire Transfer або кредитною картою Western Union. Виробник *TSD* також приймає міри до збільшення кількості покупців, проводячи широкі акції з великими знижками – так, покупцям пропонується знижка у 50% від

вартості програми за перехід на *TSD* з іншої системи дистанційної інсталяції програмного забезпечення, діє знижка у 30% для освітніх та некомерційних установ.

Підсумовуючи аналіз *TSD* можна дійти висновку, що це – вузькоспеціалізована програма з дуже багатим функціоналом, який дозволяє обирати різні способи сканування мережі та встановлення корпоративних додатків. Разом з потужністю, дане розмаїття можливостей створює істотне навантаження на адміністратора та містить приховані ризики для безпеки комп'ютерів абонентів.

1.2.2. Програма «Maestro AutoInstaller»

Іншим наявним рішенням є програма *Maestro AutoInstaller* - програма для автоматичної установки програмного забезпечення за заданим шаблоном, на різних комп'ютерах.

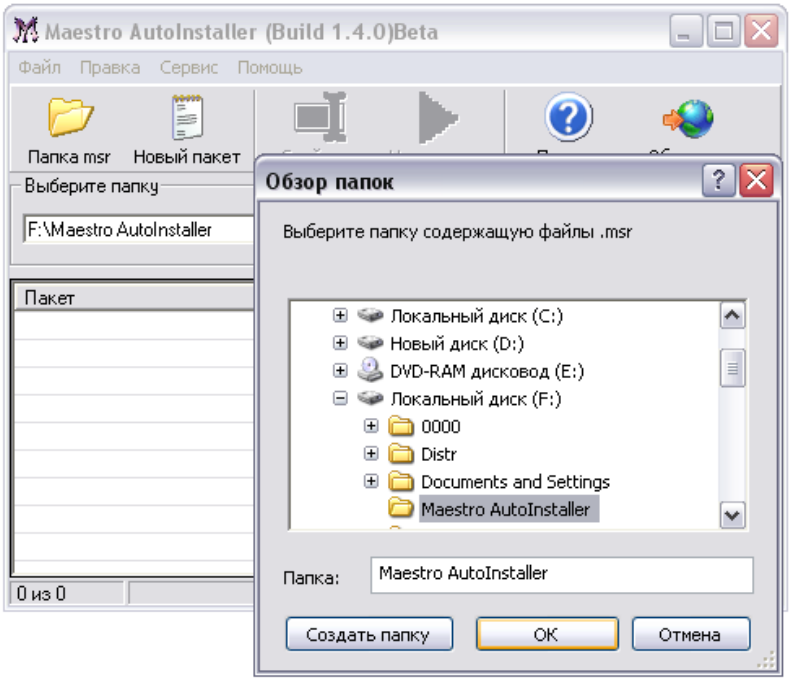

Рисунок 1.2. Інтерфейс програми Maestro AutoInstaller

На відміну від попереднього прикладу, *Maestro AutoInstaller* є програмою що від початку розрахована на дії в рамках однієї ОС та працює по принципу «запам'ятовування» дій користувача у ході інсталяції, створюючи сценарії встановлення програмного забезпечення за його діями та виконуючи потім автоматичне встановлення програм по даним сценаріям. Це означає, що для використання програми як автоматизованої системи дистанційної інсталяції потрібно проводити налаштовування доступу до файлів програми як на сервері, так і на клієнті, та, відповідно, наявності встановленої *Maestro AutoInstaller* однакової версії на обох комп'ютерах. Через це програму не можна повною мірою розглядати як чисту автоматизовану систему дистанційної інсталяції – для використання *Maestro AutoInstaller* потрібно підготовити усі комп'ютери мережі, встановивши на них одинакові версії цієї програми. Суттєвим недоліком цієї програми також є схильність програми вступати в конфлікт з технологію User Access Control (UAC), що використовується в операційних системах Microsoft Windows починаючи з ОС Windows Vista – забезпечення нормальної роботи *Maestro AutoInstaller* потребує спеціального відключення контролю облікових засобів користувачів через панель керування безпекою операційної системи, що тягне за собою загрозу потенційного порушення безпеки доступу до ресурсів Windows.

Принцип роботи *Maestro AutoInstaller* грунтується на складанні скриптів – намічених сценаріїв інсталяції. Цей принцип схожий на режим «Макро» програми Total Software Deployment, відрізняючись від нього меншим рівнем автоматизованості. Користувач має власноруч обрати (або створити) директорію, у яку будуть зберігатися скрипти інсталяції. Після цього користувач створює новий інсталяційний пакет, проводить контрольну інсталяцію програмного забезпечення на свій комп'ютер у режимі запису, формуючи скрипт, який потім додається до пакету та

відправляється на комп’ютери – абоненти мережі. При інсталяції слід мати на увазі, що програма не відслідковує подвійного клацання мишею, коліщатка миші і натискань клавіш клавіатури.[16] Можливе створення прообразів інсталяційних завантажувальних дисків за допомогою вбудованого майстра (команда Файл => Майстер створення диска). Подібний прообраз записується на жорсткий диск і включає файл autorun і три папки: папку з програмою Maestro AutoInstaller, папку зі сценаріями автоустановки і папку з збірками встановлюваних додатків. Прообраз записують на CD / DVD-диск через встановлену на комп'ютері програму для запису дисків. Слід мати на увазі, що доведеться вручну правити шляху до виконуваних файлів кожного проекту - тобто після запису прообразу на перезаписуваний диск потрібно змінювати властивості у кожного пакета, вказуючи конкретні файли на CD / DVD-диску. Одночасно можна запустити лише один екземпляр програми Maestro AutoInstaller. [16] Також програма підтримує режим «тихої інсталяції». Її принцип загалом аналогічний програмі *Total Software Deployment*, з різницею у знов таки нижчому рівні автоматизації процесу. Для роботи програми потрібна встановлена бібліотека msvbvm60.dll. Без неї програма працювати не буде, що є суттєвим недоліком при роботі з ОС, які не є версіями ОС Windows XP.

Графічний інтерфейс програми є максимально простим, навіть мінімалістичним. Це зумовлене тим що багато операцій виконується «за лаштунками» користувачем.

Найбільшою перевагою програми над *Total Software Deployment* є її безкоштовність – як показало дослідження автора роботи, компанія, яка випускає *Maestro AutoInstaller* отримує основний прибуток з розміщення реклами, тому ця програма є freeware та доступна для завантаження без реєстрації та SMS.

Загалом *Maestro AutoInstaller* можна охарактеризувати як дешеву альтернативу *Total Software Deployment*, автоматизовану систему дистанційної інсталяції «для бідних» - дана програма є набагато простішою за свій пропієтарний аналог . Але, разом з тим, *Maestro AutoInstaller* не диспонує такою кількістю різноманітних опцій, а користувач має виконувати багато операцій зі створення директорій та оперування командами «за лаштунками», не через безпосередньо графічний інтерфейс.

Підсумовуючи результати порівняльного аналізу застосунків для автоматичної дистанційної інсталяції програмного забезпечення у корпоративних мережах, можна дійти наступних висновків:

– На сьогоднішній день різноманітними розробниками програмного забезпечення реалізовані достатньо потужні додатки для автоматичного віддаленого встановлення софту для корпоративних систем та локальних мереж;

– Ці програмні додатки, такі як Maestro AutoInstaller та Total Software Deployment повною мірою відповідають вимогам до точності та швидкості дії подібних систем, а також різноманіття доступних користувачу функцій;

– Разом з тим, ці програми не відповідають вимогам до простоти у обслуговуванні та користуванні, а у деяких випадках (Maestro AutoInstaller) потребують, окрім інсталяції, додаткової підготовки до використання;

– Через це все ще відчувається суттєва потреба у простому та ефективному програмному засобі автоматичної інсталяції, який не буде потребувати складної підготовки до використання.

Ґрунтуючись на цьому, ми бачимо потребу в розробці засобу автоматичної віддаленої інсталяції, актуального для використання у

корпоративних мережах підприємств та інших локальних комп’ютерних мережах та системах для збільшення ефективності науково-дослідного, виробничого та управлінського процесу шляхом спрощення роботи адміністраторів даних систем. Основною ідеєю розробки має стати агрегація переваг простоти у обслуговуванні та користуванні, відсутності додаткової підготовки до використання а також ефективності по швидкодії, а також позбавлення від найбільш виражених недоліків існуючих програмних рішень, таких як ускладненість у використанні та платність застосування, що дозволить вести мову про більш вигідний для користувача програмний продукт.

2 АРХІТЕКТУРА ДОДАТКУ

2.1 Аспекти та базові механізми побудови рішення в межах даного класу підсистем

Перш ніж розпочати розробку програмного забезпечення, потрібно визначити детальний архітектурний план застосунку.

Архітектура програмного забезпечення (software architecture) — спосіб структурування програмної або обчислювальної системи, абстракція елементів системи на певній фазі її роботи. Система може складатись з кількох рівнів абстракції і мати багато фаз роботи, кожна з яких може мати окрему архітектуру. Архітектура програмного забезпечення стосується фундаментальних структур програмної системи та дисципліни створення таких структур і систем. Кожна структура містить програмні елементи, відносини між ними та властивості як елементів, так і відносин. Архітектура програмної системи - це метафора, аналогічна архітектурі будівлі. Вона функціонує як концепція системи та проекту, що розробляється, викладаючи завдання, необхідні для виконання проектними командами.[1]

На даний момент на ринку рішень в області архітектури мережевого програмного забезпечення домінують три основні підходи до побудови архітектури мережевих програмних додатків: використання клієнт-серверної архітектури, веб-орієнтованої архітектури та сервіс-орієнтованої архітектури.

Сервіс-орієнтована архітектура (SOA) - модульний підхід до розробки програмного забезпечення, заснований на використанні розподілених, слабо пов'язаних (англ. loose coupling) замінних компонентів, оснащених стандартизованими інтерфейсами для взаємодії за

стандартизованими протоколами. По суті, SOA можна звести до кількох ідей, причому архітектура не диктує способи їх реалізації:

- Сполучуваність додатків, орієнтованих на користувачів.
- Багаторазове використання бізнес-сервісів.
- Незалежність від набору технологій.
- Автономність (незалежні еволюція та масштабованість).

SOA - це набір архітектурних принципів, що не залежать від технологій і продуктів, зовсім як поліморфізм або інкапсуляція.

Таким чином, системи, засновані на SOA, можуть бути незалежні від технологій розробки і платформ (таких як Java, .NET і т. Д.). Наприклад, сервіси, написані на C #, що працюють на платформах .Net і сервіси на Java, що працюють на платформах Java EE, можуть бути з однаковим успіхом викликані загальним складовим додатком. Програми, що працюють на одних платформах, можуть викликати сервіси, що працюють на інших платформах, що полегшує повторне використання компонентів.[2]

Архітектура не прив'язана до якої-небудь визначеної технології. Она може бути реалізована з використанням широкого спектру технологій. Головне, що відрізняє SOA - це використання сервісів з чіткими визначеними інтерфейсами.

Таким чином, система, заснована на SOA, може бути незалежною від технологій розробки та платформи (таких як Java, .NET і т. Д.).

SOA може підтримувати інтеграцію і консолідацію операцій в складі складних систем, однак SOA не визначає і не надає методологій або фреймворків для документування сервісів.

У деяких аспектах SOA можна розглядати як архітектурну еволюцію, а не як революцію. Він захоплює багато кращих практик попередніх архітектур програмного забезпечення.

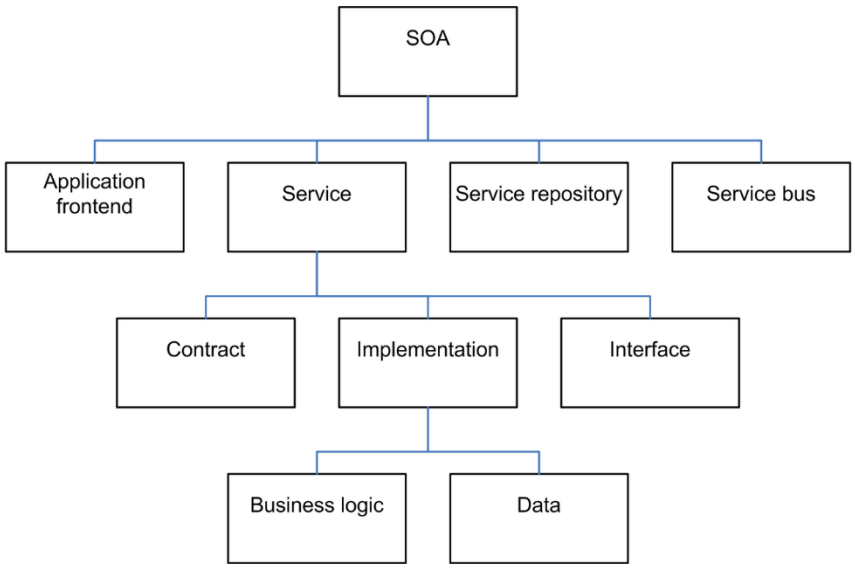

Рис. 2.1. Елементи сервіс-орієнтованої архітектури

Разом з тим, SOA має і значні недоліки, такі як:

– Незалежність від місця розташування: клієнтський код не має поняття, чи є виклик локальним або віддаленим. Звучить непогано, але тривалість затримки та види збоїв можуть сильно варіюватися. Якщо ми не знаємо, який у нас виклик, то програма не може вибрати відповідну стратегію обробки викликів методів, а значить, і генерувати вилучені виклики всередині циклу. В результаті вся система працює повільніше.

– Складна, роздута і неоднозначна специфікація: її зібрали з декількох версій специфікацій різних вендорів, тому (на той момент) вона була роздутою, неоднозначною і важкою в реалізації.

– Заблоковані канали зв'язку (communication pipes): використовуються специфічні протоколи поверх TCP / IP, а також специфічні порти (або навіть випадкові порти). Але правила корпоративної безпеки та файрволи часто допускають HTTP-з'єднання тільки через 80-й порт, блокуючи обміни даними.

Розширенням SOA є ***веб-орієнтована архітектура.*** Це стиль архітектури програмного забезпечення, який поширює сервісно-

орієнтовану архітектуру (SOA) на веб-додатки. WOA спочатку була створена багатьма веб-додатками та веб-сайтами, такими як соціальні веб-сайти та особисті веб-сайти. Ключова відмінність SOA від WOA полягає у використанні REST API.[2]

WOA має п'ять основних обмежень інтерфейсу:

– Ідентифікація ресурсу, наприклад, єдиний ідентифікатор ресурсу

– Маніпуляція ресурсами за допомогою веб-представлень, таких як HTTP

– Самописові повідомлення типу MIME

– Нейтральність програми, тобто додаток / сервіс, створений на WOA, можна розгорнути / використовувати на будь-якій платформі

Іншим можливим рішенням питання архітектури може слугувати ***клієнт-серверна архітектура***, яка міцно завоювала собі місце в області веб-розробки. Характеристика клієнт-сервер описує взаємозв'язок програм, що співпрацюють у додатку. Серверний компонент надає функцію або послугу одному або багатьом клієнтам, які ініціюють запити на такі послуги.

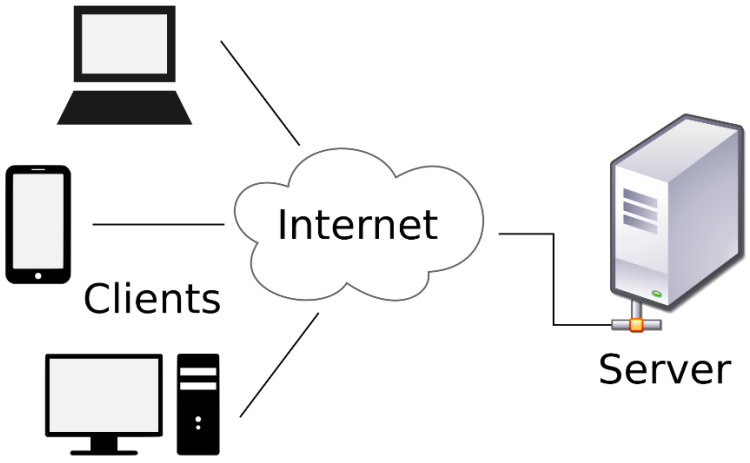

Рисунок 2.2. Модель архітектури «клієнт-сервер»

Модель «клієнт-сервер» - це розподілена структура додатків, яка розбиває завдання або робочі навантаження між постачальниками ресурсу або послуги, що називаються серверами, і запитувачами послуг, що називаються клієнтами. Часто клієнти та сервери спілкуються через комп'ютерну мережу на окремому апаратному забезпеченні, але і клієнт, і сервер можуть проживати в одній системі. Хост сервера запускає одну або декілька серверних програм, які діляться своїми ресурсами з клієнтами. Клієнт не ділиться жодним із своїх ресурсів, але він вимагає вмісту чи послуги від сервера. Таким чином, клієнти ініціюють сеанси зв'язку із серверами, які очікують на вхідні запити. Прикладами комп'ютерних програм, які використовують модель клієнт-сервер, є електронна пошта, мережевий друк та всесвітня павутина.[3]

Чи є комп'ютер клієнтом, сервером чи обом, визначається характером програми, яка вимагає функцій обслуговування. Наприклад, один комп'ютер може одночасно запускати веб-сервери та програмне забезпечення для файлового сервера, щоб обслуговувати різні дані клієнтам, які роблять різні запити. Клієнтське програмне забезпечення також може спілкуватися з серверним програмним забезпеченням у межах одного комп’ютера. Комунікація між серверами, наприклад синхронізація даних, іноді називається міжсерверною або серверно-серверною комунікацією.

Клієнти та сервери обмінюються повідомленнями за схемою обміну повідомленнями на запит та відповідь. Клієнт надсилає запит, а сервер повертає відповідь. Цей обмін повідомленнями є прикладом міжпроцесорної комунікації. Для спілкування комп'ютери повинні мати загальну мову, і вони повинні дотримуватися правил, щоб і клієнт, і сервер знали, що їх очікувати. Мова та правила спілкування визначені в протоколі зв'язку. Всі протоколи клієнт-сервер працюють на рівні додатків. Протокол рівня додатків визначає основні шаблони діалогу. Щоб ще більше

формалізувати обмін даними, сервер може впровадити інтерфейс програмування додатків (API). API - рівень абстракції для доступу до послуги. Обмежуючи спілкування певним контентом, це полегшує аналіз. Абстрагуючи доступ, це полегшує обмін даними між платформами. Сервер може отримувати запити від багатьох різних клієнтів за короткий проміжок часу. Комп'ютер може виконувати лише обмежену кількість завдань у будь-який момент та покладається на систему планування для визначення пріоритетності вхідних запитів клієнтів для їх розміщення. Для запобігання зловживань та максимальної доступності серверне програмне забезпечення може обмежити доступність клієнтів. Відмова від службових атак призначена для використання зобов'язання сервера обробляти запити, перевантажуючи його надмірною швидкістю запитів. Шифрування слід застосовувати, якщо конфіденційна інформація повинна передаватися між клієнтом і сервером.[3]

Клієнт-серверна архітектура є базисною для побудови більшості програмних додатків. Вона є класичною для побудови наступних типових програм:

– Web-сервери – спочатку надавали доступ до гіпертекстових документів по протоколу HTTP (Huper Text Transfer Protocol). Зараз підтримують розширені можливості, зокрема роботу з бінарними файлами (зображення, мультимедіа тощо).

– Сервери додатків - призначені для централізованого вирішення прикладних завдань в деякій предметній області. Для цього користувачі мають право запускати серверні програми на виконання. Використання серверів додатків дозволяє знизити вимоги до конфігурації клієнтів і спрощує загальне управління мережею.

– Сервери баз даних - сервери баз даних використовуються для обробки запитів користувачів на мові SQL. При цьому СУБД знаходиться на сервері, до якого і підключаються клієнтські програми.

– Файл-сервери - файл-сервер зберігає інформацію у вигляді файлів і представляє користувачам доступ до неї. Як правило файл-сервер забезпечує і певний рівень захисту від несанкціонованого доступу.

– Поштові сервери - надають послуги з відправлення та одержання електронних поштових повідомлень.

У одноранговій мережі два або більше комп'ютерів (одноліток) об'єднують свої ресурси та спілкуються в децентралізованій системі. Рівні - це рівносильні, або рівнопотужні вузли в неієрархічній мережі. На відміну від клієнтів у мережі клієнт-сервер або клієнт-черга-клієнт, однолітки спілкуються один з одним безпосередньо. У одноранговій мережі, алгоритм протоколу зв'язку одноранговий зв’язок врівноважує навантаження і навіть однолітки зі скромними ресурсами можуть допомогти поділити навантаження. Якщо вузол стає недоступним, його спільні ресурси залишаються доступними до тих пір, поки пропонують його інші однолітки. В ідеалі вузлу не потрібно досягати високої доступності, оскільки інші, зайві однолітки компенсують будь-який час простою ресурсів; у міру зміни доступності та завантаженості одноліткі протокол перенаправляє запити.

Перевагами клієнт-серверної архітектури є:

– Збереження інформації. Ведення бази даних здійснює сервер бази даних, що дозволяє забезпечити незалежність обробки даних в базі від програм користувача. Цілісність інформації підтримується централізованої обробкою конфліктів, що виникають при одночасній модифікації одних і тих же даних з різних робочих станцій.

– Стійкість до збоїв. Збій при роботі клієнта не позначається на цілісність даних і їх доступності для інших клієнтів.

– Масштабованість (здатність до розширення). Система здатна адаптуватися до зростання кількості користувачів і збільшення обсягу бази даних без заміни програмного забезпечення, а, в основному, за рахунок нарощування апаратних засобів.

– Велика захищеність інформації від несанкціонованого доступу. Захистити інформацію на сервері бази даних легше, так як права доступу адмініструються достатньо гнучко. При необхідності, прямий доступ може бути обмежений до певного поля таблиці або заборонений взагалі. При заборону прямого доступу звернення до таблиць здійснюється через проміжні процедури.

– Менше навантаження мережі одним користувачем, що забезпечує велику пропускну здатність мережі і можливість обслуговувати більшу кількість користувачів.

– Велика гнучкість та адаптивність системи.

На основі даних переваг можна прийти висновку, що кращим варіантом для реалізації системи дистанційної інсталяції програмного забезпечення буде побудова системи за принципами клієнт-серверної архітектури. Специфіка проектованого продукту дозволяє максимально реалізувати сильні стороні клієнт-серверної моделі архітектури. Через здатність до масштабованості та адаптивності, можлива конфігурація системи інсталяції під потреби найрізноманітніших підприємств з різною кількістю абонентів мережі підприємства та різною організацією. Зменшення навантаження на мережу є особливо важливим для мереж підприємств, рисою яких є велика кількість абонентів. Також дуже важливими є критерії безпеки – критичним фактором для підприємств, особливо фірм, є питання безпеки даних. Клієнт-серверна архітектура, завдяки гнучким політикам адміністрування та жорсткому колу клієнтів, дозволяє досить впевнено почувати себе службі безпеки підприємства. На

додачу до того ж, вивід зі строю будь-якого з клієнтів мережі не призводить до падіння її ефективності, вона може працювати, поки існує сервер. Разом з тим, WOA мережі значно менше підходять для розробки додатку для підприємства через відсутність чіткої ієрархії структури, що неприпустимо для організованих підприємств, та через великі проблеми з безпекою у WOA-мережах, що можуть стати легшою здобиччю для хакерських атак, ніж системи, побудовані за принципами клієнт-серверної архітектури.

Згідно зі сформованими у розділі 1 вимогами до побудови системи віддаленої інсталяції програмного забезпечення, проектована система повинна мати чітко виділені клієнтську та серверну частину з ієрархічно детермінованими функціями обробки та розгортання інсталяційних пакетів. Система також повинна бути захищеною від зовнішніх втручань та бути розрахованою на використання на декількох комп'ютерах.

Усе це дає підстави обрати для побудови системи дистанційної інсталяції програмного забезпечення саме клієнт-серверну архітектуру як найбільш відповідну поставленим задачам.

2.2 Особливості побудови застосунку

Побудова клієнт-серверного застосунку відповідно до вимог, визначених у розділі 1, має свої особливості, зумовлені специфічним характером використання системи дистанційної інсталяції.

Використання систем такого типу на мережах підприємства створює потребу забезпечення високого рівня безпеки від побічних втручань у роботу системи. Втручання можуть бути як і природними (завади та помилки при передачі даних) так і штучними (хакерські атаки типу «man in the middle»). Ціллю таких атак може стати крадіжка даних, втручання до роботи системи або просте порушення комунікації між клієнтом та сервером

з метою ускладнення роботи системи. Вирішити цю проблему можна або шляхом екранування приміщення для виключення впливу завад усіх типів, окрім механічних, або шляхом модифікації математичного алгоритму системи в бік збільшення його надійності. Перший шлях потребує значних матеріальних затрат, тому кращим варіантом буде використання випробуваного криптографічного методу захисту та автентифікації файлів, що передаються мережею. Найбільш розповсюдженим методом є метод хешування файлів алгоритмом MD5.

Хеш-функція - це будь-яка функція, яка може бути використана для відображення даних довільного розміру у значення фіксованого розміру. Значення, повернені хеш-функцією, називаються хеш-значеннями, хеш-кодами, дайджестами або просто хешами. Значення використовуються для індексації таблиці фіксованого розміру, яка називається хеш-таблицею. Використання хеш-функції для індексації хеш-таблиці називається хешуванням або розсіянням адреси зберігання.

MD5 (англ. Message Digest 5) - 128-бітний алгоритм хешування, розроблений професором Рональдом Л. Рівестом з Массачусетського технологічного інституту (Massachusetts Institute of Technology, MIT) в 1991 році. Призначений для створення «відбитків» або дайджестів повідомлення довільної довжини і подальшої перевірки їх достовірності. Широко застосовувався для перевірки цілісності інформації та зберігання хешів паролів. На рисунку 2.3. зображена схема однієї операції даного алгоритму. MD5 складається з 64 цих операцій, згрупованих у чотири раунди з 16 операцій. F - нелінійна функція; одна функція використовується в кожному раунді. Mi позначає 32-бітний блок введення повідомлення, а Ki позначає 32-бітну константу, різну для кожної операції. <<< s позначає обертання лівого біта на s місцями. S варіюється для кожної операції.

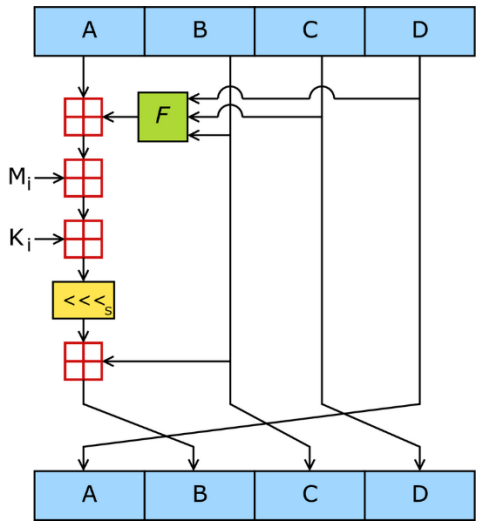

Рисунок 2.3 – схема роботи алгоритму MD5

В рамках роботи системи автоматичної дистанційної інсталяції програмного забезпечення даний алгоритм працює, створюючи хеш-суму для інсталяційного пакету, що передається мережею від сервера до клієнта. Хеш-сума передається клієнту, при отриманні інсталяційних файлів проводиться розрахунок хеш-функції для отриманого файлу. Якщо у ньому існують найменші відмінності від вихідного, хеш-сума вийде відмінною, що буде означати пошкодження файлу. При цьому інсталяція програми буде автоматично блокована системою для запобігання розповсюдження ушкодження.

Також використання у рамках мережі підприємства робить необхідною можливість одночасної інсталяції програм на декілька клієнтів. Це можна достигнути класичним методом надання системі автоматичної дистанційної інсталяції ознаки багатопоточності. Багатопоточність (англ. Multithreading) - властивість платформи (наприклад, операційної системи, віртуальної машини та т. ін.) або додатку, що складається в тому, що процес, породжений в операційній системі, може складатися з декількох потоків, що виконуються «паралельно», то є без запропонованого порядку в часі. При

виконанні деяких завдань такий поділ може досягти більш ефективного використання ресурсів обчислювальної машини. [4]

До переваг багатопоточної реалізації тієї чи іншої системи можна віднести наступні:

– Спрощення програми в деяких випадках, за рахунок винесення механізмів чергування виконання різних слабо взаємопов'язаних підзадач, що вимагають одночасного виконання, в окрему підсистему багатопоточності.

– Підвищення продуктивності процесу за рахунок розпаралелювання процесорних обчислень і операцій введення-виведення.

В рамках проектованої системи використання багатопоточності буде означати, що для процесу взаємодії між клієнтом та кожним новим сервером має створюватися новий системний потік виконання, виділений ексклюзивно для конкретного клієнта. На практиці це означає, що для встановки програми на один клієнт буде створюватися один системний потік, на два – два потоки, на десять – десять потоків і так далі. Кількість можливих потоків обмежена лише можливостями комп’ютера-сервера та кількістю абонентів у локальній мережі підприємства.

2.3 Технічні рішення

На сьогоднішній день для розробки застосунків типу «клієнт-сервер» на ринку IT-послуг застосовуються різноманітні технології, з яких найбільш поширеними є Java, .NET та Node.js. Розглянемо кожну з поточних технолоій та опишемо переваги і недоліки реалізації веб-додатків.

.NET Framework - програмна платформа, випущена компанією Microsoft в 2002 році. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), яка підходить для різних мов

програмування. Функціональні можливості CLR доступні в будь-яких мовах програмування, що використовують цю середу. [3]

Програма для .NET Framework, написана на будь-якій підтримуваній мові програмування, спочатку перекладається компілятором в єдиний для .NET проміжний байт-код Common Intermediate Language (CIL) (раніше називався Microsoft Intermediate Language, MSIL). У термінах .NET це називається збірка, (англ. Assembly). Потім код або виповнюється віртуальною машиною Common Language Runtime (CLR), або транслюється утилітою NGen.exe в виконуваний код для конкретного цільового процесора. Використання віртуальної машини переважно, оскільки позбавляє розробників від необхідності піклуватися про особливості апаратної частини. У разі використання віртуальної машини CLR вбудований в неї JIT-компілятор «на льоту» (just in time) перетворює проміжний байт-код в машинні коди потрібного процесора. Сучасна технологія динамічної компіляції дозволяє досягти високого рівня швидкодії. Віртуальна машина CLR також сама піклується про базову безпеку, управління пам'яттю і системні винятки, позбавляючи розробника від частини роботи.[9]

Архітектура .NET Framework описана і опублікована в специфікації Common Language Infrastructure (CLI), розробленої Microsoft і затвердженої ISO і ECMA. У CLI описані типи даних .NET, формат метаданих про структуру програми, система виконання байт-коду і багато іншого.[4]

Об'єктні класи .NET, доступні для всіх підтримуваних мов програмування, містяться в бібліотеці Framework Class Library (FCL). У FCL входять класи Windows Forms, ADO.NET, ASP.NET, Language Integrated Query, Windows Presentation Foundation, Windows Communication Foundation та інші. Ядро FCL називається Base Class Library (BCL).

Загалом, серед переваг .NET Framework можна перелічити наступні:

– .NET підтримує C #, C ++, VB.NET, PHP, Python, Ruby та інші мови програмування.

– Вбудований веб-сервер

– Він підтримує Microsoft Unit Testing Framework і NUnit для тестування одиниць і використовує послуги розробки NET для сценарію веб-сервера.

– Доступність даних дозволена через ADO.NET та OLeDB.

– Працює в операційній системі Windows і має ASP.NET MVC та Spring.NET як основу веб-додатків.

– Серверні компоненти включають NET і COM, тоді як компоненти GUI - клас .NET.

– Він працює в процесі поєднання CLR з рантаймом.

Програмна платформа Java - ряд програмних продуктів і специфікацій компанії Sun Microsystems, раніше незалежної компанії, а нині дочірньої компанії корпорації Oracle, які спільно надають систему для розробки прикладного програмного забезпечення та вбудовування її в будь-яке крос-платформенне програмне забезпечення. Java використовується в найрізноманітніших комп'ютерних платформах від вбудованих пристроїв і мобільних телефонів в нижньому ціновому сегменті, до корпоративних серверів і суперкомп'ютерів у вищому ціновому сегменті.

Технологія Java-аплетів стала рідко використовуваної в настільних комп'ютерах, проте вона іноді використовується для поліпшення функціональності і підвищення безпеки при перегляді всесвітньої павутини.

Програмний код, написаний на Java, віртуальна машина Java виконує байт-код Java. Однак є компілятори байт-коду для інших мов програмування, таких як Ada, JavaScript, Python, і Ruby. Також є кілька нових мов програмування, розроблених для роботи з віртуальною машиною Java. Це такі мови як Scala, Clojure and Groovy. Синтаксис Java в основному

запозичений з Сі і С ++, але об'єктно-орієнтовані можливості засновані на моделі, використовуваної в Smalltalk і Objective-C [1]. В Java відсутні певні низькорівневі конструкції, такі як покажчики, також Java має дуже просту модель пам'яті, де кожен об'єкт розташований в купі і всі змінні об'єктного типу є посиланнями. Управління пам'яттю здійснюється за допомогою інтегрованої автоматичної збірки сміття, яку виконує JVM.

Node.js - це крос-платформене середовище виконання JavaScript з відкритим кодом, яке виконує JavaScript за межами веб-браузера. Node.js дозволяє розробникам використовувати JavaScript для написання інструментів командного рядка та для сценаріїв на стороні сервера - запуску скриптів на стороні сервера для створення динамічного вмісту веб-сторінки до відправки сторінки у веб-браузер користувача. Отже, Node.js представляє парадигму "JavaScript усюди", що об'єднує розробку веб-додатків навколо однієї мови програмування, а не різних мов для скриптів на сервері та клієнтах.

Хоча .js є стандартним розширенням імені файлу для коду JavaScript, ім'я "Node.js" не посилається на конкретний файл у цьому контексті, а є лише назвою продукту. Node.js має керовану подією архітектуру, здатну до асинхронного вводу / виводу. Ці варіанти дизайну мають на меті оптимізувати пропускну здатність та масштабованість у веб-додатках із багатьма операціями вводу / виводу, а також для веб-додатків у режимі реального часу (наприклад, програм зв’язку в режимі реального часу та браузерних ігор).

Node.js дозволяє створювати веб-сервери та мережеві інструменти за допомогою JavaScript та колекцію "модулів", які керують різними основними функціональними можливостями. Модулі передбачені для вводу / виводу файлової системи, мереж (DNS, HTTP, TCP, TLS / SSL або UDP), бінарних даних (буферів), функцій криптографії, потоків даних та інших

основних функцій. Модулі Node.js використовують API, призначений для зменшення складності програм серверного запису.

JavaScript є єдиною мовою, яку Node.js підтримує у базовій версії, але доступно багато мов компіляції в JS. Як результат, програми Node.js можна записати в CoffeeScript, Dart, TypeScript, ClojureScript та інші.

Node.js використовується в основному для створення мережевих програм, таких як веб-сервери. Найбільш істотна відмінність Node.js від PHP полягає в тому, що більшість функцій у блоці PHP до завершення (команди виконуються лише після завершення попередніх команд), тоді як функції Node.js не блокують (команди виконуються одночасно або навіть паралельно, і використовувати зворотні дзвінки для сигналізації завершення чи відмови).

Node.js офіційно підтримується в Linux, macOS та Microsoft Windows 8.1 та Server 2012 (і пізніших версіях) з підтримкою рівня 2 для SmartOS та IBM AIX та експериментальною підтримкою для FreeBSD. OpenBSD також працює, і версії LTS доступні для IBM i (AS / 400). Наданий вихідний код також може бути побудований на аналогічних операційних системах, які офіційно підтримуються або змінюються третіми сторонами для підтримки інших, таких як NonStop OS та серверів Unix.

Порівняння даних технологій проводиться за наступними ключовими критеріями: ресурсозатратність, швидкодія, використання оперативної пам’яті комп’ютера, зручність коду до прочитання та складність реалізації.

За ресурсозатратністю .NET має певну перевагу над іншими двома технологіями завдяки наявності підтримки делегатів (вказників), що слугують у якості методів, які можуть бути викликані без завдання цільового об’єкту. У Java та Node.js для досягнення такої ж функціональності потрібно використовувати інтерфейси з одним методом або інший засіб обходу, що може призвести до зростання займаного

програмою місця через написання невизначеної кількості додаткового коду у залежності від додатку.

За швидкодією усі три технології мають приблизно однакові характеристики, але .NET має перевагу при роботі з ОС Windows через безпосередню пов'язаність з CLI. Більш того, у випадку недостатньості швидкодії код може бути замінений кодом будь-якої з мов, що підтримують стандарт .NET.

За складністю реалізації .NET також має перевагу завдяки своїй більшій загальності та інтегрованості під ОС Windows. Node.js, зі свого боку, є значно більш вимогливою мовою через широке використання декларативного програмування на відміну від імперативного програмування, якого потребують мови .NET.

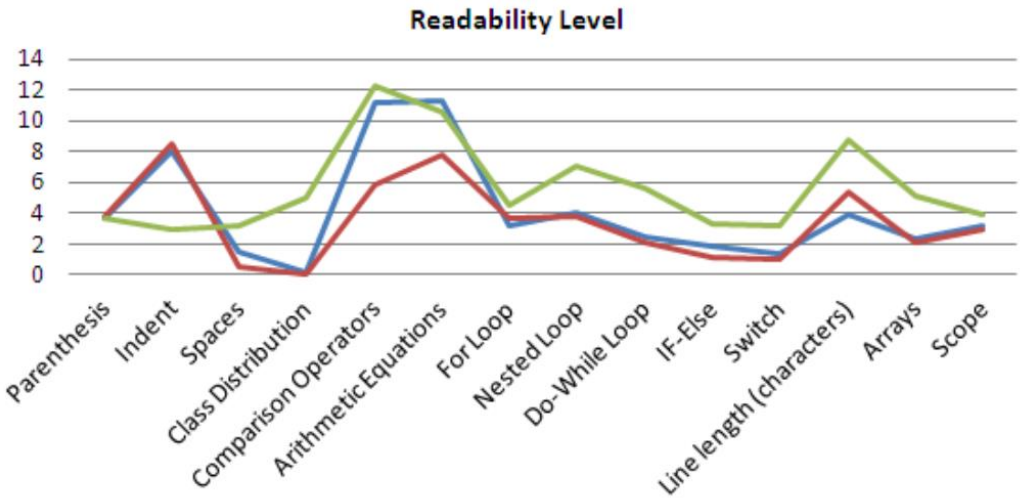

Рисунок 2.4. Графік рівня зручності читання коду C# .NET, Java та Node.js

На рисунку 2.4. зображено графік залежності рівня зручності для читання коду, написаного за стандартами технологій .NET (мова C#, колір - червоний), Java (синій) та Node.js (зелений). Як видно з графіка, в цілому найбільш виграшний для програміста з цієї точки зору C#, код якого є

простішим за код Node.js в усіх сферах окрім парентезисів та загалом таким ж простим як код Java, але одночасно виграючим у сфері арифметичних рівнянь та операторів порівняння.

Отже, .NET є оптимальною для виконання поставленого завдання технологією за совокупністю цих переваг, а також через факт переваги комп'ютерів, що використовують ОС Windows практично в усіх сферах суспільства.

Підбиваючи підсумки, можна стверджувати, що проектована система дистанційної інсталяції програмного забезпечення буде ієрархічною клієнт-серверною програмною системою, що складається з чітко виділеного сервера та багатьох клієнтів – абонентів мережі підприємства. Реалізована дана архітектура має буди засобами технології .NET (мова програмування C#) з диверсифікованими по клієнтським та серверним модулям логіками обробки бізнес-процесів. Система повинна реалізовувати взаємодію між сервером та клієнтами (що полягає у встановленні з'єднання між сервером та клієнтами, встановленні, оновленні та деінсталяції програмного забезпечення, розірванні з'єднання між сервером да клієнтами) маючи при цьому властивість багатопотоковості, уможливлюючи тим самим одночасну взаємодію меж сервером та багатьма клієнтами одночасно що зробить можливою безперебійну поставку інсталяційних пакетів до адресатів. Пакети повинні бути захищені від зламу та пошкодження методами md5-шифрування, що дозволить відмінити встановлення програмного забезпечення у випадку неспівпадіння контрольної та результуючої хеш-сум інсталяційного пакету, що буде сигналізувати про його пошкодження.

3 ВЛАСНА РЕАЛІЗАЦІЯ

Розроблене програмне рішення в межах аргументованої клієнт-серверної архітектури складається з наступних компонентів

- Серверного додатку;
- Клієнтського додатку
- Міжсервісної комунікації.

Серверна частина додатку відповідає за централізовану відправку пакетів програмних засобів на набір клієнтів. В відповідності до обраного протоколу взаємодії в межах між сервісної комунікації реалізація передачі інформації здійснюється TCP/IP протоколом.

Клієнтська частина додатку відповідає за отримання набору даних від сервера та процес встановлення пакету програмного забезпечення в прихованому (Silent) режимі.

Кожен з наведених вище компонентів в межах реалізації на обраній технології містить свої особливості . а також є шаблонні рішення в межах архітектури. Оскільки програмний засіб працює з мережею виникає проблема перевірки коректності доставки пакетів між клієнтом та сервером. Враховуючи вагу інсталяційних файлів, можливе стороннє навантаження на мережу необхідно реалізувати спосіб перевірки та перенаправлення файлів. Наступним пунктом важливо зазначити, що в випадку передачі великого обсягу інформації наявне навантаження на мережу, та внесення коректив в розмір пакету кількість каналів відносно нього. Наступним елементом огляду є завдання по прихованому встановленню пакетів програмного забезпечення, оскільки дана операція повинна виконуватись автоматично, необхідно позбавити користувача програмного засобу необхідності переходити від компютера до компютера, та впливати на процес установки. Розглянемо наявні рішення даних завдань.

3.1. Шаблоні рішення

3.1.1. Верифікація достовірності отриманих файлів

Причина виникнення даного завдання полягає в логічній поведінці програмного засобу та обмеженнях які на неї накладаються. Оскільки програмний засіб є розподіленим та складається з клієнтської та серверної частини і існує рівень мережевої взаємодії зявляється ризик появи мережевих втрат через ряд критеріїв:

1. Погане з’єднання
2. Низька швидкість мережі
3. Навантаженість мережі.

Відповідно необхідно оцінити ризики, які є наслідком втрати пакетів, в межах цілісного відображення результуючого елементу, що передається.

Наприклад, для передачі звукового контенту допускається втрата частини пакетів в межах 5% - 7%, оскільки це не значним чином впливає на якість відтворення.

В випадку логічної поведінки даного програмного засобу контент, що передається – інсталяційні пакети програмного забезпечення. В випадку не цілісного інсталяційного файлу можуть відбутись колізії в процесі встановлення, або файл в цілому може не виконатись, що призведе до псування реєстру операційної системи, збиткових записів файлів інсталяційних пакетів на диску. Враховуючі можливі критерії виникнення ризиків передачі та навантаження на мережу, що виникатиме в наслідок роботи додатку дані ризики є суттєвими і критичними для достовірної роботи програмного засобу. Відповідно виникає необхідність перевірки достовірності файлу в межах відсутності мережевих втрат. Оскільки передається пакет інсталяційних файлів необхідно перевірити цілісність пакету та в випадку не достовірної передачі перевірити дочірні файли

інсталяційного пакету виявивши помилки та перенаправити тільки їх. Для реалізації такої оцінки необхідно отримати унікальний ідентифікатор, що базується на кількості байтів всього інсталяційного пакету та окремо взятого інсталятору. Завдання даного класу вирішуються за допомогою хеш функцій, що будують хеш суму по тим чи іншим значенням. Оскільки мова йде про певний обсяг байтів, яким характеризується поточний файл та сформований пакет інсталяційних файлів хеш суму необхідно брати по ним. Для вирішення даного завдання використовувалась хеш функція MD5, аргументація вибору якої та алгоритм її роботи наведено в огляді технічних рішень.

3.1.2. Розподілення навантаження на мережу та оптимізація алгоритму

Логічна поведінка програмного рішення передбачає передачу значного обсягу інформації, на значну кількість кінцевих клієнтів через мережу.

В відповідності до даної вимоги виникає завдання розподіленої передачі файлів – багато канальної передачі та завдання балансування навантаження на мережу. Для вирішення завдання розподіленої передачі інформації було прийняте рішення про створення механізму мультиканальної передачі інсталяційних пакетів. Дане рішення включає в себе створення локального ресурсу системи - потоку який в свою чергу в межах виконуваного методу відкриває мережеве з'єднання на основі TCP/IP протоколу отримує дескриптор мережевого потоку та виконує буферизованний запис байтів окремо взятого інсталяційного файлу. Даний підхід вирішує завдання оптимізації по критерію швидкодії процесу відправлення даних між сервером та клієнтами, проте створює збиткове навантаження на мережу, що негативно відображається на між мережевій взаємодії інших застосунків. Відповідно постає завдання балансування створюваного

навантаження на мережу, а також виходячи з принципів розробки багатопоточних програмних засобів, а саме з концепції роботи багатопотчності в ОС виникає обмеження на одночасне використання потоків – їх кількість не повинна перевищувати кількість наявних логічних потоків процесору. Було прийняте рішення про введення програмного обмеження на доступний пул потоків паралельного виконання, тобто розмір наявного пулу для всіх каналів рівний кількості логічних потоків конкретної машини на якій запущено серверний компонент системи. Для оптимізації навантаження на мережу відбувається перевірка наявності клієнту в тій самій мережі чи під мережі, що містить один рівень вкладеності відносно даної. Відповідно в момент запуску серверного додатку виконується перевірка навантаженності мережі і швидкості передачі на основі чого формується коєфіцієнт від 0 до 1. На основі даного коєфіцієнту формується відносна оцінка якості з'єднання, що впливає на кількість каналів що будуть створені в момент передачі інформації, що оновлюється до моменту початку передачі даних. Дане рішення забезпечує адаптацію програмного застосунку до мережі в якій він знаходиться, що покращує інтеграцію програми до стеку програмного забезпечення, що використовується компанією.

3.1.3. Автоматичне визначення наявних клієнтських додатків у мережі

Оскільки програмний засіб працює в межах мережі та розрахований на значну кількість клієнтів на які потрібно встановити пакети програмного забезпечення, а маршрутизація відбувається за допомогою IP протоколу, то користувачеві необхідно задавати або проміжок IP адрес, або вказувати їх вручну, що не є зручним для кінцевого користувача, постає завдання автоматичного визначення клієнтів в межах мережі. Для вирішення даного

завдання було використано UDP протокол та його можливість broadcast, а саме в момент запуску клієнтський додаток надсилає пакет через broadcast на всі IP адреси, що знаходяться в межах однієї мережі, сервер в свою чергу отримує дане повідомлення та спробує встановити TCP з'єднання з клієнтом, після чого клієнт додається в список наявних клієнтів, що доступні для відправки інсталяційних пакетів (рис 3.1.)

Рисунок 3.1. – перелік доступних клієнтів в мережі

3.1.4. Встановлення в прихованому режимі та побудова інсталяційних пакетів

Виходячи з поставленого завдання потрібно реалізувати механізм встановлення програмного забезпечення без втручання користувача – автоматично, а також забезпечити коректну версію ПО в відповідності до версії ОС. Серверний компонент зберігає інформацію про кожного клієнта яку клієнт направляє в вигляді першого повідомлення до серверу, що містить версію ОС, та інформацію по набору характеристик, таких як кількість логічних потоків та максимальну частоту роботи кожного, об'єм

оперативної пам’яті, кількість вільного місця на диску з ОС, після встановлення мережевого з’єднання. В відповідності до чого серверний компонент формує пакет інсталяторів в відповідності до заданих обмежень. Дані обмеження є результатом пошуку мінімальних системних вимог до популярних програмних засобів та їх різних версій в відповідності до сумісності з ОС. Кожен окремо взятий інсталятор програмного забезпечення передбачає мануальне втручання користувача. Тому необхідно забезпечити виконання даного процесу автоматично. Для вирішення даного завдання використовуються так звані параметри автоматичного (silent) встановлення, що унікальні для кожного інсталяційного файлу. Відповідно було проведено пошук та зібрано словник даних аргументів для більше ніж 100 популярних інсталяцій для різних версій програмних продуктів в відповідності до версії ОС. Даний перелік наведений в додатку Б.

3.2. Інтерфейс міжмережевої взаємодії

В відповідності до обраної архітектури в межах проектування програмного засобу, виділяється компонент між сервісної взаємодії. Виходячи з огляду протоколів наведеного в межах аргументації існуючих рішень було прийняте рішення про використання TCP/IP протоколу, для забезпечення між компонентної взаємодії. Будь-яка операційна система містить підтримку мережевого інтерфейсу взаємодії – сокету (Socket). В межах технології розробки програмних рішень існують службові класи, що забезпечують реалізацію взаємодії через компонент системи та безпосередньо являють собою клас обгортку над ресурсом системи. Загальний вигляд роботи сокетів наведено на рис. 3.2.

В процесі мережевого з'єднання два процеси обмінюються даними. Сокет (socket) представляє собою абстрактну точку мережевого з'єднання, а якщо говорити більш конкретніше, то сокет, є кінцевою точкою мережевих комунікацій. Кожен сокет, що використовується, має тип і асоційований з ним процес. Сокети існують у середині комунікаційних доменів. Домени це абстракції, які мають конкретну структуру адресації і безліч протоколів, які визначають різні типи сокетів у середині домена. Прикладами комунікаційних доменів можуть бути: UNIX домен, Internet домен, і т.д.

У Internet домені, сокет – це комбінація IP адреси і номера порту, який однозначно визначає окремий мережевий процес у всій глобальній мережі Internet. Два сокети, один для хост-одержувача, інший для хост-відправника. Вони визначають з'єднання для протоколів, орієнтованих на встановлення зв'язку, таких, як TCP.

У UNIX інтерфейс сокетів був вбудований в систему. Інтерфейс Winsock не входить до складу Windows, а реалізований у вигляді динамічно завантаженої бібліотеки DLL.

До переваг Winsock можна віднести наступне:

– Перенесення вже існуючого коду, написаного для Berkeley Sockets API, здійснюється безпосередньо.

– Системи Windows легко вбудовуються в мережі, що використовують, як версію Ipv4 протоколу TCP/IP, так і версію IPv6, що поступово розповсюджується.

– Сокети можуть використовуватися спільно з введенням/виведенням Windows, що перекривається, що, окрім всього іншого, забезпечує можливість масштабування серверів при збільшенні кількості активних клієнтів.

– Сокети можна розглядати, як дескриптори (типу HANDLE) файлів при використанні функцій ReadFile і WriteFile, та з деякими

обмеженнями, при використанні інших функцій. Ця можливість виявляється зручною в тих випадках, коли потрібне використання асинхронного введення/виведення і портів завершення введення/виведення.

– API Winsock містить набір функцій. Специфікація Winsock розбиває їх на три групи:

– функції сокетів в стилі Берклі, включені в Winsock API;

– функції для роботи з базами даних, що дозволяють програмам отримувати інформацію про імена доменів, протоколи і т.д.

– функції, що розподіляють набір функцій інтерфейсу сокетів Берклі.

Крім того, всі функції можна розділити на дві великі групи: блокуючі і не блокуючі.

Блокуюча функція примушує програму, що викликала її, чекати закінчення мережевої операції введення/виведення. Як приклад приведемо декілька функцій цієї групи:

– accept – підтверджує запит на встановлення з'єднання. Утворює новий сокет і сполучає його з видаленим комп'ютером, що запрошує з'єднання. Початковий сокет повертається в стан прийому вхідних запитів.

– closesocket – закриває одну сторону в з'єднанні сокетів;

– connect – встановлює з'єднання на вказаному сокеті;

– recv – приймає дані із сполученого сокета;

– send – передає дані через сполучений сокет;

– sendto – передає дані через сполучений або не сполучений сокет;

Не блокуюча функція не примушує програму, що викликала її, чекати закінчення мережевої операції введення/виведення. Вони перетворюють інформацію або мають справу з локальним сокетом. Як приклад приведемо декілька функцій цієї групи:

– bind – привласнює ім'я не ініціалізованому сокету;

– getsockname – повертає ім'я вказаного локального сокета;

– getsockopt – повертає опції вказаного сокета;

– htonl – перетворює порядок байтів в 32-розрядному числі з машинно-незалежного в мережевий;

– htons – перетворює порядок байтів в 16-розрядному числі з машинно-незалежного в мережевий;

– inet_addr – перетворює рядок з IP-адресою у форматі десяткового з крапкою в 32-розрядне число в мережевому форматі;

– inet_ntoa – перетворює IP-адрес у формат десяткового з крапкою;

– listen – встановлює вказаний сокет в стан прослуховування запитів на вхідне з'єднання. Сама функція не проводить мережевих операцій введення/виведення;

– ntohl – перетворює порядок байтів 32-розрядного числа з мережевого в машинно-залежний;

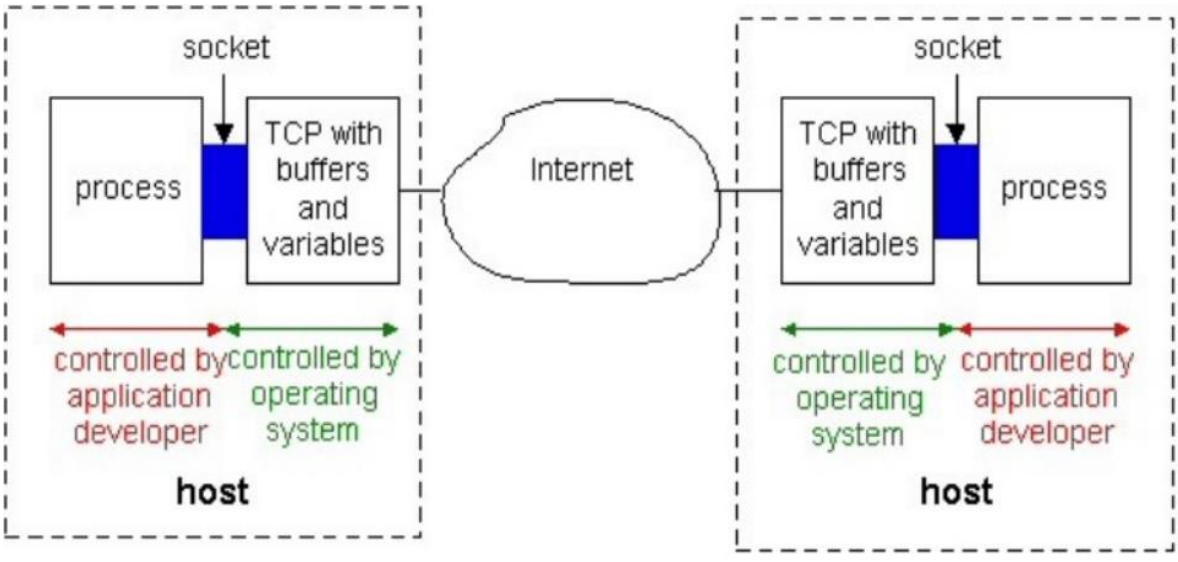

Рисунок 3.2. Принцип роботи сокетів

Робота сокету регламентується послідовним викликом на кожній стороні паралельно набору функцій. Розглянемо дану взаємодію детальніше. Оскільки протокол TCP э протоколом транспортного рівня то взаємодія між компонентами відбувається в межах даного рівня в межах моделі OSI та безпосередньої реалізації операційною системою. Детальну інформацію про точку інтеграції та міжкомпонентну взаємодію на основі моделі OSI і Windows API наведено на рис 3.3.

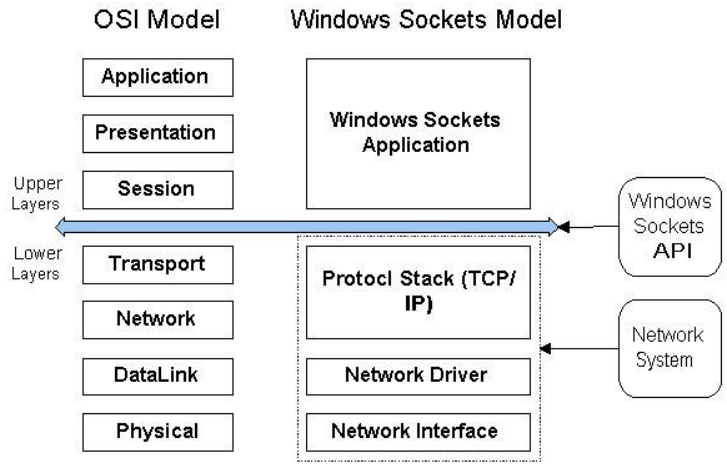

Рисунок 3.3. Схема міжкомпонентної взаємодії

Виходячи з наведеної взаємодії ОС і функцій її API та реалізацій драйверу відбувається розподілення відповідальності за передачу інформацій через мережевий ресурс. Реалізація логіки поведінки сокетів в межах ОС Windows, зосереджено в WS2_32.DLL рис 3.3. Даний елемент операційної системи відповідає за створення ресурсу формування дескриптору даного ресурсу і його реєстрацію в системі. Будь-які подальші виклики по передачі інформації в межах поточного сокету здійснюється викликом методів даної DLL з вказанням дескриптору ресурсу.

Оскільки інтерфейс Winsock повинен відповідати промисловому стандарту, прийняті у ньому угоди про правила привласнення імен і стилі програмування дещо відрізняються від тих, з якими ми звикли стикатися в

процесі роботи з функціями Windows. Слід зазначити, що Winsock API не є частиною Win32/64. Крім того, Winsock надає додаткові функції що не відповідають стандартам; ці функції використовуються лише у разі крайньої необхідності.

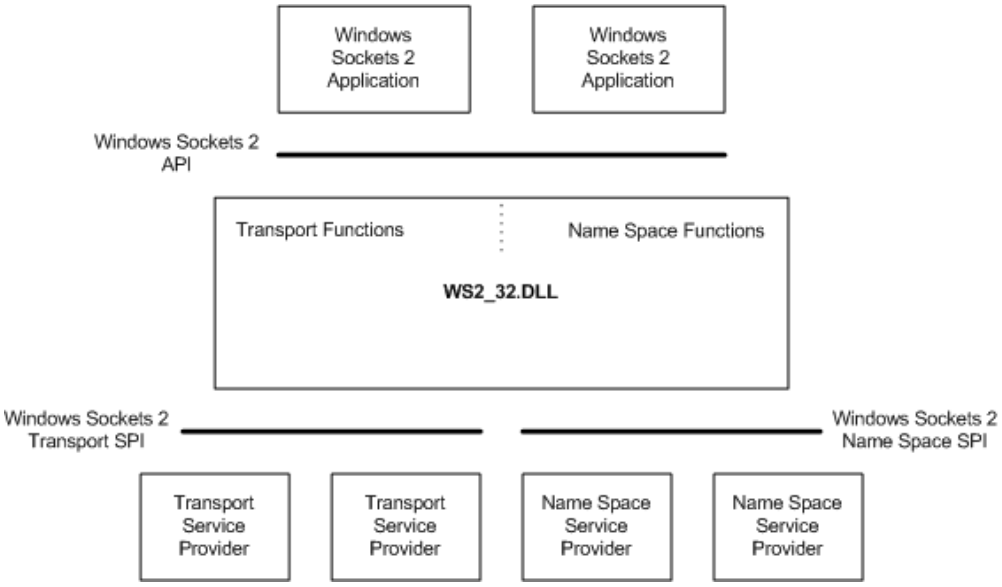

Рисунок 3.4. Схема логіки взаємодії сокетів

Прикладний програмний інтерфейс платформи Win32 має хороші можливості для створення мережених програм. В операційній системі Windows наявна бібліотека, яка надає можливість працювати з базовою технологією передачі даних – сокетами. Сокети – програмні інтерфейси взаємодії програм. На базі сокетів спроектовано величезну кількість програм для роботи з мережею та написано багато бібліотек, які слугують «обгортками» для сокетів, щоб абстрагуватися від деталей реалізації передачі даних.

Сокети побудовані на концепції «клієнт-сервер». Тобто одна машина (програма) виступає в ролі сервера, яка надає (виконує) сервіс, а інша в ролі клієнта, що дає запит серверу на виконання тих чи інших операцій. Все це добре інкапсульовано в потужних бібліотеках роботи з

мережею, які дають можливість будувати потужне програмне забезпечення. Крім того, на базі сокетів працюють найпоширеніші протоколи 7-го (прикладного) рівня моделі OSI, такі як ftp, http, smtp, pop3 і т.д.

Розглянемо функції що доступні в системі для побудови мережевої взаємодії на рівні додатку в межах сокету. Необхідно розмежувати клієнтський та серверний сокет. Для створення сокету на стороні серверу, що приймає TCP з'єднання, необхідно виконати наступні дії:

1) Створити об'єкт типу Socket, вказавши йому тип мережі (в наведеному нижче прикладі AddressFamily.InterNetwork (IPv4), тип транспортного протоколу SocketType.Stream (TCP) і ProtocolType.IP);

2) Зв'язати отриманий сокет з IP адресою і портом на сервері, викликавши метод Bind сокета. Як параметр Bind приймає об'єкт класу IPEndPoint, інкапсулює в собі IP адреса сервера і порт, до якого будуть підключитися клієнти.

3) Встановити сокет в стан прослуховування, викликавши метод Listen і передавши йому в якості параметра розмір черги чекають обробки підключень.

Для клієнтського сокету необхідно :

1. Створити об'єкт типу Socket, вказавши йому тип мережі (в наведеному нижче прикладі AddressFamily.InterNetwork (IPv4), тип транспортного протоколу SocketType.Stream (TCP) і ProtocolType.IP);

2. Зв'язати отриманий сокет з IP адресою і портом на сервері, викликавши метод Bind сокета. Як параметр Bind приймає об'єкт класу IPEndPoint, інкапсулює в собі IP адреса сервера і порт, до якого будуть підключитися клієнти.

3. Встановити сокет в стан прослуховування, викликавши метод Listen і передавши йому в якості параметра розмір черги чекають обробки підключень.

4. Необхідно створити делегат типу AsyncCallback і викликати метод BeginAccept, передавши йому в якості параметра делегат і наш слухач сокет.

5. При підключенні клієнта буде викликаний делегат, якому в властивості AsyncState параметра типу IAsyncResult прийде наш слухач сокет.

6. У отриманого сокета викликається метод EndAccept, який повертає новий об'єкт Socket, через який відбувається обмін повідомленнями з віддаленим клієнтом.

7. Знову викликається BeginAccept, цикл роботи повторюється.

Дана послідовність дозволяє сокетам доволі надійно встановлювати та підтримувати з’єднання між собою.

Алгоритм взаємодії наведено на рис. 3.5.

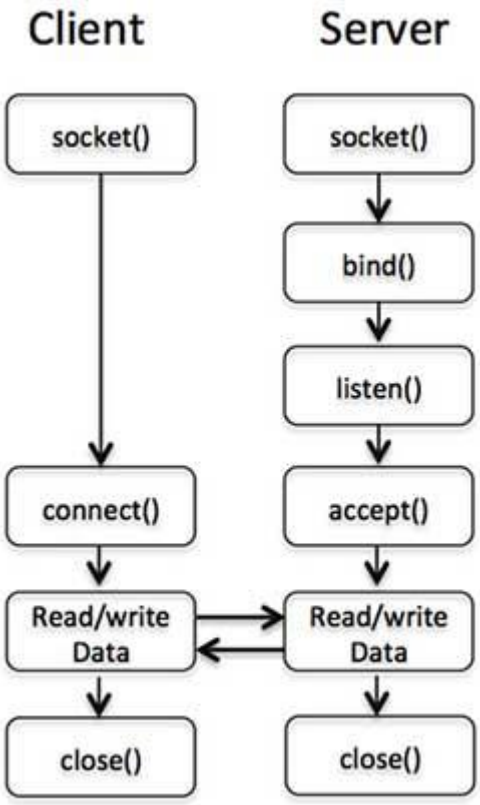

Рисунок 3.5. Алгоритм взаємодії сокетів

3.3. Серверна частина

Серверна частина програмного засобу створена з використанням технології WinForms, що забезпечує гнучкий інтерфейс для кінцевого користувача. Дана частина програмного засобу включає в себе описані шаблонні рішення. Логіку збереження інформації про клієнтів та версійність операційної системи на якій вони запущені. Відповідає за стан мережі та її навантаження. Визначає кількість допустимих потоків.

Серверна програма складається з трьох вікон: головного, що містить варіації для встановлення, програми, логи, та додаткового вікна яке містить список ip-адрес комп'ютерів на яких запущений клієнтський додаток та форми що відповідає за задання та відображення масиву ip-адрес. Форма що відповідає за масив ip-адрес містить в собі список UI-контролерів, що

включають в себе прогресбар, повідомлення, що відповідає за результат встановлення, кількість загальних програм та кількість переданих. Форма, що відповідає за виведення результатів сканування мережі вкючає в себе функцію обробник події додавання юзера та функцію перевірки на активність чек боксів, що відповідають за комп'ютери в мережі на яких запущена клієнтська частина програми (тобто є серверною частиною функції сканування мережі).

1. MainForm

Додаток використовує основні класи для роботи:

1.1. FileUtil

1.2. ErrorType

1.3. InstallerData

1.4. InstallerResult

1.5. InvalidHashException

1.6. NetworkStateChecker

Форма має основні процедури та функції для роботи:

1 SendProgramToClient

2 listIpFormUser_MySave

2. IpForm

Додаток використовує основні класи для роботи:

2.1. ServerClass

2.2. ClientOnServer

3. ListIpsFromUser

Форма має основні функції:

3.1 CreateArrayIP()

Клас FileUtil використовується для створення Хеш-коду для файлу, який буде передаватися. Далі зі сторони Клієнта буде відісланий Хеш-код файлу, який отримали на клієнті, і сервер перевіряє на схожіть Хеш-коди.

Від цього залежить чи сервер дозволить встановлення програми чи відмовить клієнту в інсталяції.

Клас ErrorType містить в собі перелічувальну константу (enum ErrorType) , що набуває одне із трьох значень :

1. None //передача файлу вдала
2. InvalidHash // передача файлу відбулася проте md5 сума отримана на сервері і клієнті не зійшлася
3. Exception // помилка іншого типу

a. і відповідає за вірність передачі файлу.

Клас InstallerData містить в собі всю інформацію про файл, що передаватиметься, а саме: ім’я файлу, аргументи для встановлення в прихованому режимі на клієнті , настройки для консольного вікна клієнту , хеш сума файлу отримана на сервері , і байти отримані з самого файлу.

Клас InstallerResult відповідає за наявність / відсутність помилки після отримання файлу , містить в собі поле з типом помилки і поле з описом даної помилки.

Клас NetworkStateChecker відповідає за запуск фонового потоку, що виконує сканування мережі та логує поточне завантаження формуючи коєфіцієнт якісної оцінки навантаження на мережу, в межах якого виконується відносна оцінка відповідно до якої регламентується кількість каналів, що створюватимуться при передачі інформації клієнтам.

Процедура CreateArrayIP, яка виконує роль створення масиву IP-адрес, по яким будуть передаватися інсталяційні файли та повертає кількість IP-адрес що знаходяться в проміжку між першою та остайньою адресою.

SendProgramToClient відповідає за передачу файлів до адресату та встановлення ПЗ на віддаленому ПК. В випадку коли використовується форма ListIpsFromUser, тобто задається масив ip-адрес, то дана функція використовує крос поточний делегат для відображення прогресу

встановлення під час безпосередньої роботи даної функції на окремому потоці.

ClientOnServer відповідає за маніпуляції з даними клієнтів (ip,port). ServerClass відповідає за отримання списку користувачів (List<ClientOnServer>) та роботу з ним, а саме генерування події при додаванні користувача. Також функціонал даного класу дає змогу по отримати UDP пакет, що надіслав клієнт в мережу, за допомогою нього дістати адрес та порт клієнту, після чого надіслати ip-адрес та порт сервера на клієнт. Вигляд головного вікна серверу приведений на рисунку 3.6. Графічний інтерфейс є інтуїтивно зрозумілим, з відображенням підключених до мережі клієнтів та доступних до інсталяції програм у розподілених віконних модулях.

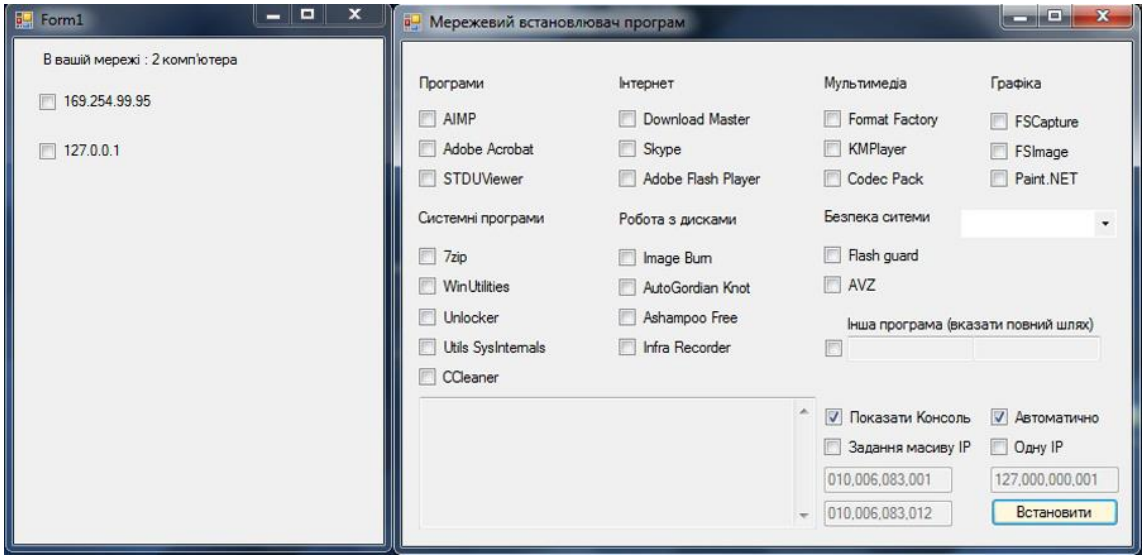

Рис. 3.6. Серверний додаток.

Після запуску серверного додатку користувач має можливість вибрати варіанти задання ip-адрес для встановлення при виборі задання масиву ip-адрес запускається форма на якій вказується перша та остання ip-адреса далі генерується список UI контролів в яких міститься прогресбар що відповідає за прогрес інсталяції, результат інсталяції та скільки із скількох програм було передано,також міститься чек бокс чи використовувати цю адресу в

випадку коли в список потрапляють вимкнені машини чи машини на які інсталювати ПЗ не потрібно, після натискання вибору потрібних адрес їх список передається на до головного вікна програми, після натискання кнопки "Встановити" відбувається багатопоточна передача даних на кожну з ір-адрес, після кожної передачі запускається крос - поточний делегат що зупиняє потік передачі даних, і відає на форму що містить список клієнтів результат встановлення, збільшує прогрес бар на 1, після завершення передачі всіх даних, та натискання Ok на повідомлення "програма завершила свою роботу" очищується значення прогресбарів

Якщо користувач поставить відмітку в чек боксі автоматично, то запуститься форма на якій буде виведено результат сканування мережі, а саме відображено всіх користувачів на яких запущений клієнтьсякий додаток, після вибору потрібних користувачів та натискання кнопки встановити відбувається багатопотокова передача інформації.

Коли користувач натискає на чекбокс "одну ір" то текстове поле стає активним вводиться потрібна ір-адреса і однопоточно передаються дані на клієнт.

Після отримання вхідних даних відбувається підключення сервера до клієнта. У випадку успішного підключення далі відбувається пересилання об’єкту (інсталяційного пакету). Після підтвердження отримання у випадку успішної передачі відбувається закриття з’єднання. Функція для передачі даних наведена на рисунку 3.7.

Рис. 3.7. Функція передачі даних на клієнт

Після запуску функції SendProgramToClient отримуємо вхідні дані та заповняємо об’єкт класу InstallerData, а саме записуємо ім’я файлу отримуємо та записуємо хеш суму, параметри встановлення та відображення консолі клієнту, зчитуємо файл в масив байтів та записуємо його, підключаємося до клієнту надсилаємо дані, а саме об’єкт класу InstallerData отримуємо звіт про успішність встановлення на клієнті переданого ПЗ, та логуємо його, після чого закриваємо з’єднання.

3.4. Клієнтська частина

Додаток базується на таких основних процедурах:

- SetConsoleWindowVisibility
- GetSystemInfo
- SeekServerUdp
- Install

– Receive

Виконує роль отримувача файлу від сервера та збереженні цього файлу на локальному диску.

Функція SetConsoleWindowVisibility показує приховує вікно консолі.

Функція Install отримує в якості параметрів шлях до файлу аргументи для прихованої інсталяції (без втручання користувача) .

Функція GetSystemInfo відповідає за збір інформації в відповідності до конфігурації клієнтського додатку, станом на зараз базова конфігурація містить наступні елементи для відслідковування, версія ОС, кількість оперативної памяті, кількість вільного місця на диску, характеристики процесору.

Функція SeekServerUdp відповідає за надсилання UDP пакету в мережу та отримання адресу серверу (клієнтська частина функції сканування мережі).

Клієнтський додаток запускається у прихованому стані. Вигляд клієнтського додатку приведений на рисунку 3.8.

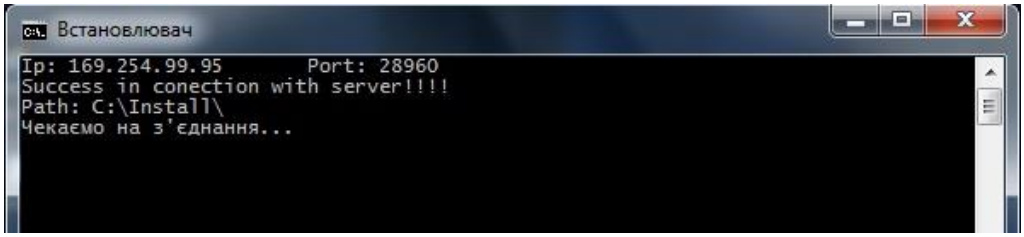

Рис. 3.8. Клієнтський додаток.

Після запуску , отримуємо ip-адресу машини на якій знаходиться клієнтський додаток, далі формуємо UDP пакет, поміщаємо в нього даний ip-адрес та відправляємо в мережу, після цього створюємо або очищуємо директорію для збереження даних що в подальшому будуть отримані від

сервера, після вдалого підключення отримуємо об’єкт класу InstallerData який містить в собі хеш-суму, отриману на сервері, параметри для прихованого встановлення та відображення консолі. У випадку якщо хеш-суми рівні - робимо спробу встановити програмне забезпечення отримане з сервера. Відповідно, якщо спроба вдала, то код помилки None передається на сервер. Якщо відбулася помилка під час встановлення, то код даної помилки передається на сервер, а якщо хеш-суми були нерівні то код помилки InvalidHash передається на сервер, після чого закривається з’єднання і програма переходить до пункту прослуховування порту.

4. ВИСНОВКИ

У ході науково-дослідних робіт над проектом автором був проведений аналіз вимог до системи дистанційної інсталяції програмного забезпечення, яка могла би полегшити роботу бізнесових та науково-учбових закладів. Визначені необхідні характеристики проектованої системи, а саме: масштабованість, високий степінь захисту та надійності, можливість установки, оновлення та деінсталяції програм з комп'ютерів – абонентів мережі підприємства у реальному часі, мінімізація навантаження на операторів абонентських комп'ютерів. Проведений аналіз наявних на момент написання роботи на ринку систем дистанційної інсталяції рішень у відповідності до сформованих вимог.

Ґрунтуючись на результатах аналізу автором роботи було намічено шлях розробки власного застосунку. Проведений огляд існуючих принципів архітектури програмних додатків, наведені їх позитивні та негативні характеристики. Засновуючись на них була обрана найбільш відповідна поставленій меті розробки архітектура, а саме клієнт-серверна архітектура програмного забезпечення, як найбільш відповідна умовам застосування проектованого додатку. Проаналізовані три найбільш поширені мови програмування, що застосовуються для побудови клієнт-серверних застосунків, а саме .NET C#, Java та Node.js. Проведене порівняння продуктивності, зручності до читання та швидкодії даних мов, в результаті чого прийнято рішення про розробку системи засобами технології .NET так мови програмування C#.

У результаті науково-дослідних робіт була спроектована система дистанційної інсталяції програмного забезпечення, що складається з клієнтського та серверного модуля і відповідає поставленим вимогам. Спроектована система дозволяє проводити вибір програм із переліку,

задання проміжку IP адрес з наявних, спостерігати за процесом встановлення, оновлення, видалення програмного забезпечення у режимі реального часу, проводити сканування мережі на наявність клієнтів. Реалізований бізнес-процес формування, обробки та багатопотокової передачі інсталяційних пакетів на всі IP адреси мережі підприємства. Навантаження на користувача знижене шляхом розподілення бізнес-процесів по відповідним застосункам клієнта та сервера. Розроблена система відповідає сучасним вимогам до UX та UI, захисту інформації. Автору вдалося розробити користувацький інтерфейс, що дозволяє користувачу використовувати можливості даного програмного продукту в повній мірі.

Спроектована система дистанційної інсталяції програмного забезпечення була протестована автором на локальній мережі зі змінною кількістю клієнтів, практично доведена працездатність даної системи та проведено порівняння її ефективності з наявними на ринку аналогами, такими як Total Software Deployment та Maestro AutoInstaller. Результати порівняння були записані та зображені у графічному вигляді. З них автором був зроблений висновок про те, що розроблений додаток хоч і не має кардинальної переваги над існуючими аналогами, але й не поступається їм у швидкодії, при цьому переважаючи їх у простоті використання та інтерфейсу.

Сучасний світ стає немислимим без комп'ютерних мереж. Запропонований автором програмний продукт дозволяє задовольнити потреби користувачів, які повсякчасно зустрічаються з необхідністю встановлення, модернізації та оновлення програмного забезпечення не тільки в офісах великих компаній, а й у локальних мережах наукових підприємств, сервісних центрів та банків, тим самим дозволяючи економити час та кошти користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Microsoft Windows Server 2003. Справочник администратора/ Пер. с англ. — М.: Издательско-торговый дом «Русская Редакция», 2003. - 640 с.:
2. Дж. Скотт Хогдал, «Анализ и диагностика компьютерных сетей» — М.: Лори, 2007. – 352 с.:
3. В. Олифер, Н. Олифер, «Компьютерные сети. Принципы, технологии, протоколы. 4-е издание.» 4-е изд. — СПб.: Питер, 2010. — 944 с.:
4. Д. Вулстон, " Платформа .NET 2.0 для профессионалов" — "Вильямс ", 2007. – 464 с.:
5. Э. Трёльсен," C# и платформа .NET. Библиотека программиста " — СПб.: Питер, 2008. — 800 с.:
6. Э. Трёльсен " C# и платформа .NET 3.0" — М.: Лори, 2009. — 1456с.:
7. К. Нейгел, Б. Ивьен, Д.Глинн, М. Скиннер, К. Уотсон, " C# 2005 и платформа .NET 3.0 для профессионалов" — СПб: Диалектика, 2007. — 1376с
8. "Microsoft .NET Remoting" /Скотт Маклин, Джеймс Нафтел, Ким Уильямс.
9. Э. Кровчик, В. Кумар, «.NET. сетевое программирование для профессионалов», — М.: Лори, 2005. — 417 стр.
10. http://msdn.microsoft.com/uk-ua/default.aspx (дата звернення: 10.04.2020).
11. http://msdn.microsoft.com/uk-ua/vstudio/default.aspx (дата звернення: 11.04.2020).
12. Г. Шилдт, «C# 4.0: полное руководство»— М.: ООО "И.Д. Вильямс", 2011. — 1056 с.:
13. http://msdn.microsoft.com/ru-ru/library/6sh2ey19.aspx(дата звернення: 13.04.2020).
14. http://msdn.microsoft.com/ru/library/system.runtime.serialization.formatters.binary.binaryformatter.aspx (дата звернення: 11.04.2020).
15. http://msdn.microsoft.com/ru-ru/library/system.net.sockets.tcpclient.aspx (дата звернення: 12.04.2020).
16. http://maestro-kit.ucoz.ru/ (дата звернення 01.11.2019).